Thèse

THÈSE INSA Rennes
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le grade de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Joseph CHAZALON
ÉCOLE DOCTORALE : MATISSE
LABORATOIRE : IRISA – UMR6074

Interprétation contextuelle et assistée de fonds d'archives numérisés : application à des registres de ventes du XVIII^{e} siècle

Thèse soutenue le 9 janvier 2013
devant le jury composé de :

Jean-Marc OGIER
Professeur à l'université de La Rochelle / *Président et rapporteur*

Josep LLADÓS
Associate Professor à l'UAB (Espagne) / *Rapporteur*

Rolf INGOLD
Professeur à l'université de Fribourg (Suisse) / *Examinateur*

Christopher KERMORVANT
Responsable *R&D* chez A2iA / *Examinateur*

Jean CAMILLERAPP
Professeur à l'INSA de Rennes / *Directeur*

Bertrand COÜASNON
Maître de conférence (HDR) à l'INSA de Rennes / *Encadrant*

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Interprétation contextuelle et assistée de fonds d'archives numérisés: application à des registres de vente du XVIIIe siècle

Nom Prénom de l'auteur : CHAZALON JOSEPH

Membres du jury :
- Monsieur CAMILLERAPP Jean
- Monsieur OGIER Jean-Marc
- Monsieur COUASNON Bertrand
- Monsieur INGOLD Rolf
- Monsieur LLADOS Josep
- Monsieur KERMORVAN Christopher

Président du jury : OGIER Jean-Marc

Date de la soutenance : 09 Janvier 2013

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse ne pouvant être reproduite
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 09 Janvier 2013

Signature du président de jury

Le Directeur,

M'hamed DRISSI

Remerciements

J'ai eu l'immense chance de rencontrer, au cours de mon doctorat, beaucoup de personnes admirables dont les conseils, les avis, ou l'exemple, m'ont aidé à progresser. Dans la mesure où ce document est destiné à un public scientifique, et plus particulièrement à la communauté du traitement de documents, je ne nomme ici que celles et ceux dont les recherches s'inscrivent dans ce domaine.

Je tiens à remercier chaleureusement les personnes suivantes :

Les membres de mon jury Ces chercheurs reconnus m'ont fait l'honneur de juger mes travaux. MM. LLADÓS et OGIER ont accepté les rôles de rapporteurs, et MM. INGOLD et KERMORVANT ceux d'examinateurs.

Mes directeurs de recherches doctorales Grâce à leur confiance et leurs conseils, MM. CAMILLERAPP et COÜASNON, m'ont, quant à eux, transmis leur vision novatrice du problème d'interprétation de documents, ainsi que, je l'espère, la rigueur de leur raisonnement.

Mes collègues au sein de l'équipe Intuidoc Au cours des différents projets auxquels j'ai participé, j'ai eu la chance de travailler avec Mme LEMAITRE-LEGARGEANT, ainsi que MM. GUICHARD et TOSELLI, pour la mise en œuvre de nos contributions. J'ai également eu l'occasion d'échanger longuement avec MM. ALMAKSOUR, DELAYE et MACÉ, dont les qualités scientifiques et humaines sont incontestables.

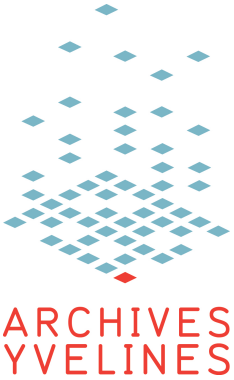
Mes collègues et amis à l'IRISA L'Institut de Recherche en Informatique et Systèmes Aléatoires est un cadre idéal pour réaliser un doctorat. Les collègues et amis que j'y ai rencontrés m'ont donné l'opportunité de discuter simplement de mon approche, des difficultés à surmonter, mais aussi de toutes autres sortes de choses plus légères. Ces échanges m'ont considérablement aidé, tant professionnellement que personnellement.

Mes collaborateurs associatifs Les différents collaborateurs au sein des services académiques, des entreprises, ou des associations de jeunes chercheurs, avec qui j'ai travaillé à l'élaboration de projets associatifs, ont été pour moi des exemples par leur professionnalisme.

Ma famille et mes amis proches Finalement, mes proches, famille et amis, ont su rester présents et me soutenir au cours de ces années bien chargées.

Financement

Ce travail a été effectué en collaboration avec les Archives Départementales des Yvelines, en France, avec le support du Conseil Général des Yvelines.



Résumé

Les fonds d'archives forment de grandes quantités de documents difficiles à interpréter automatiquement : les approches classiques imposent un lourd effort de conception, sans parvenir à empêcher la production d'erreurs qu'il faut corriger après les traitements.

Face à ces limites, notre travail vise à améliorer la processus d'interprétation, en conservant un fonctionnement page par page, et en lui apportant des informations contextuelles extraites du fonds documentaire ou fournies par des opérateurs humains.

Nous proposons une extension ciblée de la description d'une page qui permet la mise en place systématique d'échanges entre le processus d'interprétation et son environnement. Un mécanisme global itératif gère l'apport progressif d'informations contextuelles à ce processus, ce qui améliore l'interprétation.

L'utilisation de ces nouveaux outils pour le traitement de documents du XVIII^{e} siècle a montré qu'il était facile d'intégrer nos propositions à un système existant, que sa conception restait simple, et que l'effort de correction pouvait être diminué.

Mots-clés

– analyse et reconnaissance de documents
– interaction homme-machine
– interaction asynchrone
– fonds d'archives
– interprétation itérative

Références de l'auteur

Les idées et figures présentées dans cette thèse furent partiellement introduites, au préalable, dans les publications listées ci-après.

Conférences internationales

Conférences avec actes et comités de relecture

- **Chazalon, J.** ; Coüasnon, B. & Lemaitre, A. (2012), A Simple And Uniform Way To Introduce Complimentary Asynchronous Interaction Models In An Existing Document Analysis System, *in* '10th IAPR International Workshop on Document Analysis Systems (DAS'12)'.
- **Chazalon, J.** & Coüasnon, B. (2012), Iterative analysis of document collections enables efficient human-initiated interaction, *in* 'Proc. of SPIE-IS&T Electronic Imaging, Document Recognition and Retrieval XIX', SPIE.
- **Chazalon, J.** ; Coüasnon, B. & Lemaitre, A. (2011), Iterative Analysis of Pages in Document Collections for Efficient User Interaction, *in* 'International Conference on Document Analysis and Recognition (ICDAR'2011)', pp. 503–507.
- Guichard, L. ; **Chazalon, J.** & Coüasnon, B. (2011), Exploiting Collection Level for Improving Assisted Handwritten Words Transcription of Historical Documents, *in* 'International Conference on Document Analysis and Recognition (ICDAR'2011)', pp. 875–879.
- **Chazalon, J.** & Coüasnon, B. (2010), Using definite clause grammars to build a global system for analyzing collections of documents, *in* Laurence Likforman-Sulem & Gady Agam, ed.,'Document Recognition and Retrieval XVII', SPIE.

Conférences nationales

Conférences avec actes et comités de relecture

- **Chazalon, J.** ; Coüasnon, B. & Lemaitre, A. (2012), Comment introduire simplement et uniformément 2 modes d'interaction asynchrones complémentaires dans un systême d'analyse de documents existant, *in* 'Conférence Internationale sur l'Ecrit et le Document (CIFED'12)', pp. 285–299.
- **Chazalon, J.** ; Coüasnon, B. & Lemaitre, A. (2010), Mémoire visuelle pour l'analyse grammaticale de documents, *in* 'Actes du XIème Colloque International Francophone sur l'Ecrit et le Document (CIFED'10)', pp. 343–354.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les documents manuscrits, imprimés et maintenant électroniques produits jour après jour par les individus et les organisations humaines sont des témoins des innombrables activités de notre époque et de celles du passé. Depuis quelques décennies, on a pu constater la diminution du coût du matériel informatique, l'augmentation conjointe de la vitesse des communications et des capacités de stockage, et surtout de nombreux progrès techniques. Tous ces éléments ont permis de transformer progressivement notre façon d'échanger grâce à la précision et la rapidité des communications électroniques. Les supports papiers se heurtent à cette exploitation informatique de leurs contenus, il devient nécessaire de les dématérialiser pour pouvoir les intégrer à ces nouveaux flux d'information.

L'exploitation dématérialisée de documents a de nombreux avantages. Si les documents sont des chèques, ou des factures, alors il devient possible de réaliser automatiquement des transactions. S'il s'agit de livres, il devient possible de rechercher un terme précis dans une bibliothèque virtuelle en quelques instants. Certains projets sont alors titanesques, comme le projet Google Book qui a permis la numérisation de 20 millions d'ouvrages entre 2004 et 2012 [50] afin d'indexer leurs contenus. Des initiatives comme celles de la Bibliothèque Nationale de France visent à rendre accessible un nombre plus modeste de contenus (quelque centaines de milliers de documents, tout de même), mais avec des objectifs différents [1]. Dans le cadre de ces *bibliothèques numériques*, il s'agit de :

- favoriser la *diffusion* de la connaissance, en permettant un accès simultané et à distance des contenus ;
- *protéger* les œuvres fragiles et rares, en autorisant une consultation non destructive, éventuellement assortie d'une restriction de visualisation de certaines parties confidentielles (cas de documents nominatifs, par exemple) ;
- permettre une *exploitation riche* des contenus, à des fins de recherche scientifique, en permettant une navigation facile entre les contenus, la recherche rapide d'éléments, ou encore la mise en correspondance entre certaines parties : index, références, etc.

Sans cette capacité à s'orienter dans le volume immense d'information mis à disposition, ces initiatives de numérisation seraient d'un intérêt très limité.

La production des informations indispensables à une exploitation riche de fonds documentaires requiert une interprétation (au moins partielle) de ces derniers afin de transcrire les informations présentées sous *forme visuelle* vers une *représentation informatique*. Il est possible, dans une certaine mesure, de réaliser automatiquement cette tâche.

Les approches actuelles suivent un processus assez linéaire schématisé comme suit par Baird *et al.* [6].

1. Numérisation Les documents sont photographiés ou scannés page par page et stockés

1. La charte du projet Gallica est disponible à l'adresse suivante : `http://www.bnf.fr/fr/professionnels/anx_pol_num/a.Charte_documentaire_de_Gallica_1997_2007.html`.

sous un format numérique (en général compressé avec perte pour limiter l'espace de stockage) avec une structure (hiérarchie informatique) minimaliste et peu de métadonnées (organisation en répertoire selon les cotes par exemple).

2. **Traitement initial** Les images sont normalisée (luminosité, contraste, rotation...) et on leur ajoute quelques métadonnées très générales.

3. **Extraction du contenu** Selon l'utilisation prévue de la version numérique, le système procède alors à l'identification des zones, à l'analyse de la structure, et à la reconnaissance du contenu : texte manuscrit ou non, symboles, tableaux, images...

4. **Remise en forme** Dans le cas des livres électroniques, le contenu extrait est adapté aux nouveaux supports de diffusion (taille d'écran en particulier).

5. **Indexation, synthèse automatique** La construction d'index inversés permet un accès multiple et facile aux contenus originaux. Le travail de synthèse permet de fournir des vues condensées des éléments intéressant l'utilisateur dans le corpus (par exemple les images des *unes* d'un journal particulier).

6. **Diffusion** La mise en ligne des contenus est alors possible, apportant de nouvelles difficultés techniques : stockage, accessibilité... Au delà des outils de recherche et de navigation, il peut être nécessaire de permettre aux utilisateurs d'apporter certaines modifications aux contenus électroniques ainsi qu'aux métadonnées pour les corriger ou les enrichir.

Dans le cadre des travaux présentés dans cette thèse, nous nous sommes intéressés à l'étape d'extraction du contenu, telle que définie précédemment, dans le contexte d'un projet de dématérialisation de fonds d'archives en partenariat avec les archives départementales des Yvelines, en France. Grâce à ce partenaire, notre équipe de recherche dispose d'images de fonds documentaires variés, parmi lesquels un ensemble de registres de ventes de biens saisis par l'État lors de la Révolution française, qui a servi de support à nos réflexions et expérimentations. Ce fonds comporte un peu moins d'une dizaine de milliers d'images, et regroupe des images de natures assez variées, manuscrites et dégradées, pour lesquelles il faut produire une interprétation structurée. Ce problème précis est donc un cas intéressant pour s'attaquer à la classe plus large de problème d'interprétation de documents structurés incertains. Nous présenterons plus en détail ces documents d'archives, dont la figure 0.1 montre un extrait, dans le prochain chapitre, en particulier à la sous-section 1.1.2.2.

Notre travail tire sa raison d'être du constat de l'incapacité des approches actuelles à traiter efficacement ce genre de document : malgré un travail de conception rigoureux, les systèmes existants ne parviennent pas à faire face suffisamment aux dégradation physiques des documents (tâches, traversées d'encre, etc.), aux artefacts de numérisation, à la variabilité importante de l'écriture manuscrite, et aux changements occasionnels de structure. De nombreux problèmes se produisent au cours du processus d'interprétation, entraînant une cascade d'erreurs dans les opérations suivantes et mènent à des résultats de piètre qualité qui nécessitent un effort de correction important. Cet effort est tellement pesant, que pour des documents de ce type, il apparaît souvent plus économique d'envisager un traitement intégralement manuel, à condition que le volume de données soit raisonnable.

Notre travail vise précisément à dépasser cette limite : **notre objectif est de diminuer l'effort de correction requis lors de l'interprétation de fonds documentaires « difficiles » en volume important, sans augmenter la difficulté de conception du système.** En particulier, nous souhaitons conserver un traitement des fonds image par image. Pour sim-

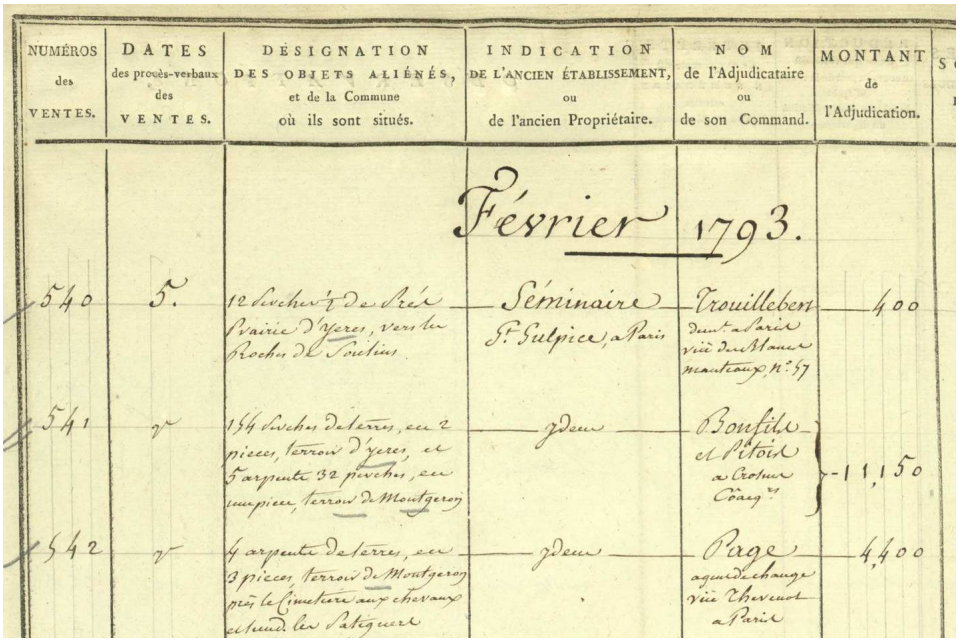


FIGURE 0.1 – Exemple de page d'un registre de ventes regroupant les informations utiles aux transferts de biens : ancien propriétaire, description du bien, nouvel acquéreur, etc.

plifier la présentation, nous considérerons dans ce manuscrit que chaque image correspond à une page.

Notre approche se base sur l'hypothèse selon laquelle les systèmes existants se priveraient de deux sources importantes de connaissances lors de l'interprétation d'images de pages isolées. La première est le *fonds documentaire* lui-même, qui donne un contexte à l'image traitée et présente de nombreuses redondances ainsi que les liens entre les contenus des pages : nous pensons qu'il est possible de tirer profit de cette redondance pour en faire un atout. La deuxième est l'*humain*, qui dispose d'une capacité exceptionnelle à interpréter les documents et qui est omniprésent au cours du cycle de vie d'un système d'interprétation automatique : depuis sa conception jusqu'à la phase de correction, tout en supervisant éventuellement les traitements.

Notre travail a alors consisté à identifier les propriétés que doit posséder un système capable de tirer profit de ces deux sources de connaissance *pendant* le travail d'interprétation, en supposant disposer d'outils satisfaisants pour la réalisation des traitements au niveau du signal (de l'image). Nous proposons une architecture permettant de mettre en place une coopération entre le programme d'interprétation d'images de page et son environnement, à savoir les autres pages du fonds documentaires et des opérateurs humains.

Nous avons travaillé à proposer des outils qui permettent d'étendre un système existant plutôt que de chercher à développer un système complet. Nous avons identifié précisément les opérations à réaliser afin de favoriser la réutilisation de nos propositions. Grâce à ces outils, un concepteur peut alors construire ou modifier rapidement et facilement un système capable de tirer profit de ces nouvelles sources de connaissance.

Grâce à des mécanismes standards qui peuvent être modélisés par des *continuations*, nous avons proposé un mécanisme de détection et de correction d'erreurs basé sur un

échange asynchrone entre l'outil d'interprétation de page et son environnement. Un mécanisme d'interprétation itératif permet la réintégration progressive de l'information externe à cette page et sa fusion avec l'information extraite de l'image, rendant possible la remise en cause de résultats précédents, le guidage du système et la progression globale de l'interprétation. Les outils proposés permettent la mise en place rapide de scénarios qui étaient trop complexes à implémenter auparavant, en exprimant simplement de nouvelles propriétés à propos des documents, et sans perturber le traitement page par page classique. Ceci permet d'apporter, lors de l'exécution, l'information utile à l'endroit précis où le système se trompe, et évite une propagation d'erreurs. La figure 0.2 illustre l'intérêt de ce fonctionnement pour un cas de sous-segmentation. Les expériences menées montrent alors, pour certaines tâches, un gain au niveau du travail de correction par rapport à une stratégie de correction à posteriori. Finalement, ces travaux ouvrent des perspectives d'amélioration progressive de tels systèmes qui étaient limitées, voire interdites, par la difficulté à produire des résultats fiables.

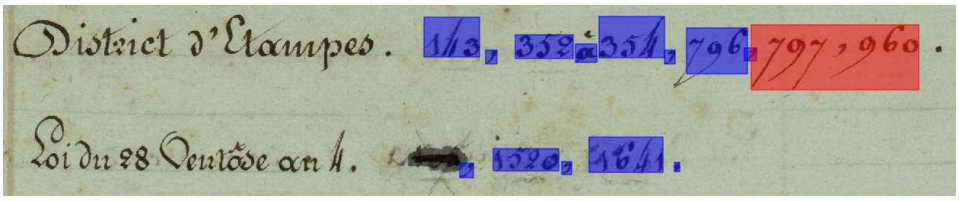


FIGURE 0.2 – Dans cette figure, les nombres et les séparateurs (en bleu) ont été correctement détectés, mais la dernière virgule de la première ligne a été manquée, entraînant un cas de sous-segmentation (en rouge). S'il est possible d'indiquer au système qu'il a raté ce séparateur et lui faire prendre en compte cette information, alors il n'est plus nécessaire de segmenter les nombres, saisir leur valeur, et réintégrer ces informations dans la structure résultat : le système s'en chargera.

La présentation de notre travail s'articule selon trois parties et une conclusion.

La partie I présente l'état de l'art Cette partie s'intéresse à extraire de l'étude des approches existantes les caractéristiques essentielles qui constituent un système d'interprétation de fonds documentaires. Il s'agit donc d'une étude complète qui justifie l'architecture et le comportement du système proposé. Sa composition est la suivante :

- le chapitre 1 donne un prolongement technique à cette introduction et montre la nécessité d'utiliser de nouvelles sources de connaissances pour permettre l'interprétation de fonds d'archives ;
- le chapitre 2 s'intéresse à la conception et à l'utilisation d'un programme d'interprétation de page, dans l'objectif de définir un système de référence que nous étendrons ensuite ;
- le chapitre 3 se focalise sur les approches permettant d'exprimer et exploiter des propriétés reliant les contenus entres plusieurs pages, afin de comprendre comment tirer profit du contexte documentaire ;
- le chapitre 4 étudie comment les approches existantes permettent de mettre en place une communication avec des opérateurs humains, et tirer profit de la connaissance dont ces derniers disposent.

La partie II décrit nos contributions Cette partie détaille progressivement la structure du système d'interprétation de fonds documentaires que nous proposons, en se basant

sur les caractéristiques et fonctionnalités extraites de l'état de l'art. Sa composition est la suivante :

- le chapitre 5 décrit l'organisation globale du système proposé et son comportement ;
- le chapitre 6 se focalise sur la manière de générer automatiquement, à partir d'une description pour laquelle nous proposons de nouveaux éléments, un programme d'interprétation de page capable de tirer profit de l'information fournie par son environnement.

La partie III valide nos propositions Cette partie montre la viabilité de nos travaux et les résultats qu'ils permettent d'obtenir. Sa composition est la suivante :

- le chapitre 7 décrit comment nous avons effectivement réalisé un système complet capable de traiter des fonds documentaires dans des conditions réelles ;
- le chapitre 8 présente les réductions de la quantité de travail manuel que nous avons mesurées lors d'expériences, et les enseignements que nous en avons tirés.

La partie IV conclue notre présentation Le chapitre 9 résume les points essentiels de nos travaux, et envisage de nouvelles pistes de recherche à partir des résultats obtenus.

Première partie

— État de l’art — Donner un environnement à l’interprétation d’images de documents

Chapitre 1

Interprétation de documents d'archives

Introduction

Ce chapitre préliminaire est un prolongement de l'introduction. Il va nous permettre d'introduire les notions essentielles de l'interprétation de documents anciens et de mieux cerner les enjeux liés au traitement de ces données.

Dans le processus de dématérialisation de documents anciens, nous nous focalisons sur l'étape d'extraction des contenus qui permet de passer d'un ensemble d'images brutes à une structure de données numériques permettant l'utilisation informatique du fonds d'archives considéré.

1.1 Caractéristiques et utilisation des documents d'archives

Comme mentionné dans l'introduction, notre travail s'est basé sur l'étude de fonds documentaires particuliers (des registres de ventes du XVIII^{e} siècle) afin de permettre une validation pratique de nos propositions, avant de pouvoir espérer les généraliser. Nous introduisons ici quelques notions et éléments de vocabulaire généraux à propos des documents d'archives.

1.1.1 Typologie documentaire

La commission de normalisation « Modélisation, production et accès aux documents » de l'Association française de normalisation (AFNOR) [2] propose de définir les *fonds d'archives* comme :

> « l'ensemble des documents de toute nature qu'une personne physique ou morale, appelée le producteur, a produits ou reçus dans l'exercice de ses activités, rassemblés de façon organique et conservés en vue d'une utilisation éventuelle. »

Le groupe de travail propose des distinguer les types de documents suivants, pouvant former tout ou une partie d'un fonds documentaire :

Manuscrits littéraires, manuscrits d'œuvre Tous les manuscrits d'une œuvre qu'elle soit littéraire ou non, publiée ou inédite.

Correspondance Lettres envoyées ou reçues par le producteur.

Archives relatives aux activités et à la vie d'une personne Documents résultants de l'activité professionnelle ou publique du producteur, comme des notes de travail ou des discours, ainsi que ceux relevant de sa vie privée, tels qu'un agenda, ou un testament.

Archives relatives à l'organisation et aux activités d'une collectivité Ensemble des documents relatifs au fonctionnement d'une collectivité, à savoir les archives juridiques, administratives, comptables et financières, techniques et scientifiques, domaniales, ainsi que la documentation et les éventuels documents relatifs à toutes autre forme d'activité.
Dans le cadre du projet qui a servi de support à notre travail de recherche, c'est cette catégorie qui nous intéresse le plus.

Collections Pièces réunies dans un but précis par un collectionneur : autographes, manuscrits, etc.

Notons que le terme *« collection »* peut également être employé pour désigner un ensemble de documents organisés selon une thématique précise, et être alors synonyme de *« fonds d'archives »*, et nous nous efforcerons d'éviter cette confusion en parlant de *« fonds d'archives »*, ou, plus généralement, de *« fonds documentaires »*.

1.1.2 Organisation documentaire

1.1.2.1 Classer pour faciliter l'accès

La commission insiste sur l'importance de l'organisation documentaire, afin de permettre l'organisation hiérarchique de fonds documentaires occupant jusqu'à plusieurs plusieurs centaines de mètres de stockage.

L'organisation des documents d'un fonds peut alors se faire selon plusieurs plans de classement, afin de décomposer progressivement la structure du fonds jusqu'à un niveau de détail satisfaisant ; la pièce, si nécessaire. Parmi ces plans, on peut trouver une organisation typologique, souvent utilisée en premier niveau, une organisation thématique, souvent utilisée pour décomposer les fonds liés à des activités, et finalement une organisation chronologique, souvent terminale.

Au niveau numérique, les archivistes veillent à maintenir ce classement après la numérisation des documents. Les images des différentes pages des documents obéissent alors à une organisation séquentielle cohérente, généralement regroupées par livre ou pièce au sein d'une structuration hiérarchique reproduisant la partie de la structure globale de laquelle le lot de documents numérisés a été extrait. Le nommage des fichiers numériques joue également un rôle important.

1.1.2.2 Exemple d'organisation : série 1Q des archives départementales des Yvelines

À titre d'exemple, on présente rapidement le contenu de la série *1Q* des archives départementales des Yvelines avec laquelle nous avons pu construire et valider notre approche. Ces documents seront utilisés abondamment dans cette thèse, à titre d'exemple. Aussi est-il pratique de les présenter dès à présent.

Par convention nationale, la série *Q* regroupe les documents relatifs à la vente des biens nationaux et des domaines de l'État (principalement) pendant la période suivant la Révolution française, entre 1791 et 1800. Chaque département possède donc un fragment de cette

série, et la numérotation présentée ci-après correspond à celle des archives départementales des Yvelines.

La série *1Q* regroupe, quant à elle, le sous-ensemble des documents relatifs aux actes de ventes. Dans le cas des fonds documentaires des archives départementales des Yvelines, on peut distinguer deux types de documents particulièrement intéressants dans ce fonds.

- 11 *registres* correspondant à 11 types de ventes, et répartis dans 329 sous-séries (cotes *1Q1* à *1Q329*) correspondant à autant de livres physiques. Chacune des pages de ces registres contient une partie d'un tableau global regroupant les informations synthétiques relatives à chaque transaction : numéro, date, localisation du bien (commune), ancien propriétaire, nouvel acquéreur, etc.
  Un extrait de page est visible en figure 1.1.
- Une *table des communes* (cote *1Q365*) qui liste, par ordre alphabétique des communes, tous les numéros des bien situés sur une commune, classés par type de vente. Cette table propose donc un index des ventes.
  Un extrait de page est visible en figure 1.2.

Nous avons pu disposer des images des pages des livres, classées par répertoire (*1Q1*, *1Q2*, etc.), numérotées dans leur ordre de lecture. Cette organisation est représentative des données dont un peut disposer lors d'une telle tâche, et montre déjà l'importance des liens entre les différentes parties d'un corpus documentaire.

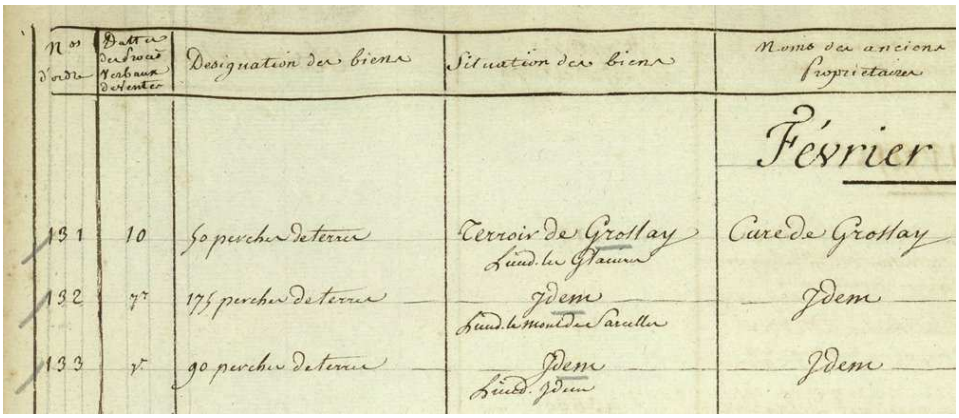


FIGURE 1.1 – Extrait d'une page de *registre* de la série *1Q*.

1.1.3 Utilisation dématérialisée de fonds d'archives

1.1.3.1 Assister le lecteur

Le principal risque dans la dématérialisation de documents est de priver le lecteur, c'est à dire l'utilisateur final du système de consultation des documents sous forme numérique, des facilités offertes par le papier. Baird [5] indique à cet effet quatre propriétés des documents sous forme papier, qu'il semble nécessaire de reproduire dans la mise en place d'une consultation dématérialisée afin de permettre une utilisation pratique et confortable :

- le papier permet une navigation flexible entre les documents ;
- le papier facilite la consultation simultanée de plusieurs documents se référençant mutuellement ;
- le papier invite à l'annotation ;

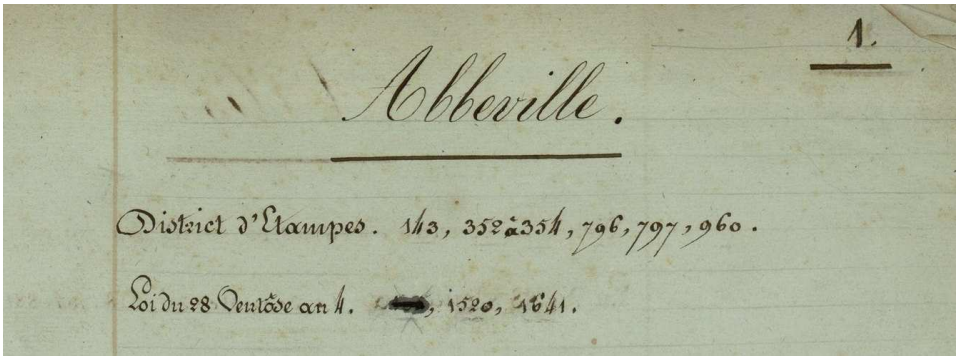


FIGURE 1.2 – Extrait d'une page de la *table des communes* de la série *1Q*.

– le papier permet l'enchevêtrement de la lecture et de l'écriture.

Faure et Vincent [38] proposent alors de centrer la conception des systèmes de consultation sur l'objectif d'assister le lecteur dans sa réappropriation du contenu des documents. Ce processus de réappropriation est alors organisé selon les quatre tâches fondamentales suivantes :

le traitement qui consiste à distinguer les signes (le contenu) du support (le fond) et saisir l'organisation des composants visuels de la page, afin de décoder le message contenu ;

la collecte qui consiste à regrouper les contenus selon des critères variés (thématiques, visuels) afin de les comparer, les retrouver, etc. ;

l'augmentation qui consiste associer des informations et des résultats de raisonnements à des contenus, c'est à dire à des parties des images des documents ;

la restructuration qui consiste à organiser l'information d'une façon différente de l'ordre de lecture original.

Assister le lecteur dans de telles opérations nécessite l'extraction d'informations riches, et la construction de structures de données adaptées à ces usages.

1.1.3.2 Transcrire et organiser

Faciliter la lecture peut être fait en proposant un nouveau formatage du document, plus adapté à l'écran du dispositif utilisé, par exemple en réagençant les blocs de contenus pour qu'ils s'affichent sur une colonne au lieu de plusieurs. Ceci présuppose de détecter l'ordre de lecture du document.

La recherche de contenus peut être facilitée grâce à la construction de plusieurs index, se superposant au plan de classement classique. S'il est possible de transcrire les contenus (cas de l'écriture imprimée en général), alors on peut imaginer de proposer au lecteur de réaliser des recherches textuelles. Toutefois, d'autres formes d'indexations sont possibles : Eglin *et al.* [36] proposent de détecter le style d'écriture de certains manuscrits pour permettre un accès par époque. On peut également imaginer des accès par auteur, ou selon tout autre élément pertinent.

Plus généralement, la navigation peut être facilitée en produisant des regroupement d'éléments similaires, c'est à dire visuellement proches. C'est une solution à la plupart des cas où les éléments ne peuvent pas être identifiés automatiquement de façon assez fiable, ni transcrits. Eglin *et al.* [36] ont mis en place un système permettant de passer en revue

les éléments visuels de manuscrits anciens (lettrines, enluminures), et Coüasnon *et al.* [27] proposent de reproduire le feuilletage en construisant dynamiquement un catalogue de mots similaires dans des registres anciens.

Coüasnon *et al.* [27] basent leur approche sur le concept d'*annotation* et proposent de distinguer :

les annotations *textuelles* contenant des données et métadonnées relatives à une image de document (date, lieu, mots-clés, etc.) ;

les annotations *graphiques* correspondant à une partie de l'image (un champs de formulaire, une cellule de tableau, une zone, etc.) représentée par un polygone ou un rectangle.

Ces deux types d'annotations peuvent être liés (se référencer mutuellement) pour permettre une description riche et flexible du contenu de la page. Pour produire ces annotations, Coüasnon *et al.* proposent un système automatique, ainsi qu'une interface de consultation qui autorise également la création et la modification d'annotations par les lecteurs.

Proposer des outils de consultation dématérialisée efficace impose donc de pouvoir identifier et interpréter les contenus pertinents dans les images des documents, et de « traduire », ou « transcrire » ces contenus dans un système de représentation qui puisse être exploité de façon informatique. La qualité de l'expérience de consultation est donc directement liée à la finesse de l'interprétation. Cette interprétation dépend de deux éléments essentiels :

- la connaissance à propos des documents ;
- l'exploitation du contexte documentaire.

1.1.4 Notion de contexte documentaire

1.1.4.1 Cas de l'écriture manuscrite

Le cas du domaine de la reconnaissance d'écriture manuscrite permet d'introduire, par similarité, la notion de langage pour les documents. Les modèles de langages sont en effet utilisés pour guider et optimiser la reconnaissance d'écriture manuscrite, et ce à différents niveaux [118, 80, 82] :

le niveau morphologique qui regroupe la connaissance à propos des différents symboles ou caractères utilisés dans une langue et de leurs fréquences ;

le niveau lexical qui regroupe la connaissance à propos des mots autorisés du langage ;

le niveau syntaxique qui regroupe la connaissance à propos de la façon d'organiser les mots selon leur catégories syntaxiques (ex : « adjectif », « verbe », etc.) dans un langage ;

le niveau sémantique qui regroupe la connaissance à propos des contraintes de cohérence au regard de la catégorie sémantique (aussi dite « fonctionnelle », ou « logique ») des mots (ex : « objet », « animal », etc.) ;

le niveau pragmatique qui permet de valider les propositions au regard des faits précédemment énoncés.

Chaque niveau réduit l'espace des énoncés corrects possibles et permet de mettre en relation les parties d'un texte pour guider, vérifier, optimiser ou même compléter son interprétation. Plus le niveau d'interprétation est élevé, plus les interdépendances entre un élément et les autres sont nombreuses. L'utilisation de connaissances sémantiques et pragmatiques dans un système de reconnaissance est difficile et rare.

1.1.4.2 Le document comme langage visuel

Il est alors possible, à l'instar du langage manuscrit, de transposer ces niveaux d'abstraction au niveau du document, voire du corpus (c'est à dire d'un ensemble de documents). C'est le constat que font Marriott et Meyer en introduction d'un ouvrage dédié à la théorie des langages visuels [74]. Un document, même textuel, peut alors comporter des zones de texte qui obéiront à un langage textuel, et on pourra considérer que les primitives visuelles, leur organisation, et les liens entre ces composants, formeront un langage visuel répondant lui aussi à des contraintes lexicales (*Quels sont les objets visuels utilisés ?* — texte, lignes, images, etc.), syntaxiques (*Comment une page est-elle organisée ?* — titres, numéros de pages, etc.), voire sémantique (*Quelles sont les contraintes entre les objets représentant les numéros de pages ?* — incrément d'une unité à chaque page) d'un langage *visuel*.

Saund [106] indique quant à lui que la validation de contraintes sémantiques (ex : la somme des prix des articles est égal au montant de la facture) ne peut être faite sans une connaissance précise de la structure du document, montrant alors l'importance d'une connaissance à priori du type de document à traiter pour extraire l'information utile.

D'autres exemples d'utilisation des liens sémantiques entre les contenus, cette fois-ci entre plusieurs pages, sont proposés par Déjean et Meunier [32] pour la reconnaissance de livres. Les auteurs exploitent certaines redondances comme celles des titres possédant une position stable en haut de page, des contraintes numériques comme l'incrémentation des numéros de pages, ou encore la cohérence dans les styles des séquences, pour optimiser l'interprétation du document.

L'exploitation de ce qu'on propose d'appeler le *contexte documentaire*, qu'il relie des contenus à l'intérieur d'une page ou entre des pages différentes, offre donc des possibilités très intéressantes qui ne sont accessibles que si on dispose d'un modèle de document, voire de corpus, suffisamment riche.

1.2 Particularités des documents d'archives pour l'interprétation

La nature des documents anciens entraîne un certain nombre de difficultés lorsqu'on tente de les interpréter, que ce soit automatiquement, ou en proposant à un humain de les lire. Nous décrivons ici les principales difficultés liées à ces données, ainsi que les éléments favorables qui peuvent permettre d'y faire face.

1.2.1 Difficultés

Cette identification des difficultés liées aux documents anciens est principalement basée sur l'analyse d'Antonacopoulos et Downton [4], ainsi que sur celle d'Ogier [86].

1.2.1.1 Ambiguïté des langages humains libres

À l'instar des autres formes de communication humaine, les messages portés par des documents produits par un humain peuvent rapidement devenir ambigus en l'absence de contraintes (connues) lors de la production. Pour illustrer cela, on peut prendre l'exemple d'une lettre dactylographiée : si l'auteur n'a indiqué qu'une adresse dans la lettre, est-ce la sienne ou celle du destinataire ? Qu'en est-il si deux adresses sont présentes ? Les positions relatives des deux blocs peuvent prêter à confusion.

Une analyse plus poussée du document peut permettre de trouver des éléments pour confirmer ou infirmer une hypothèse, mais ce n'est pas toujours possible.

1.2.1.2 Manque de connaissances à priori à cause du décalage historique

Le décalage temporel entre l'époque de création d'un document et l'époque actuelle est la première source de difficultés lors du traitement de documents anciens. Contrairement aux documents actuels, il n'est pas possible d'imposer un format que l'on maîtriserait, comme un formulaire prêt pour une lecture optique. On ne peut pas non plus profiter de notre connaissance sur les contraintes imposées par les logiciels de mise en page des documents modernes, qui produisent des contenus d'une grande régularité. Au contraire : selon les fonds, on peut disposer d'une connaissance très variable sur les conditions et les contraintes de productions des documents.

Cette méconnaissance des techniques d'écriture, des outils utilisées, du vocabulaire, de la forme des lettres, des règles de présentation et autres usages perdus font qu'il est difficile de connaître les types de contenus et les organisations de ces derniers qui pourront être rencontrés lors de l'analyse. On peut citer le cas des patronymes, qui peuvent être obsolètes voire inconnus. Le décalage historique est donc une deuxième source d'ambiguïté dans les interprétations des documents.

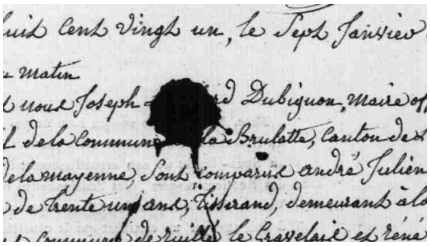
Le problème d'amorçage de le connaissance d'un système de traitement de documents est alors délicat et coûteux, comme noté par Saidali *et al.* [105]. Ceci peut être renforcé par le besoin de faire appel à un expert du fonds documentaire considéré. Le transfert de connaissances (modèles de documents, modèles de graphèmes, lexiques, etc.) est par ailleurs délicat entre différents fonds documentaires qui présentent assez peu de points communs en général.

L'élaboration de modèles permettant de guider l'analyse et l'interprétation de fonds documentaires anciens est donc un processus coûteux et difficile, qui mène généralement à la production de connaissances fragmentaires à cause du décalage historique et de l'effet de l'échantillonnage de l'ensemble des données.

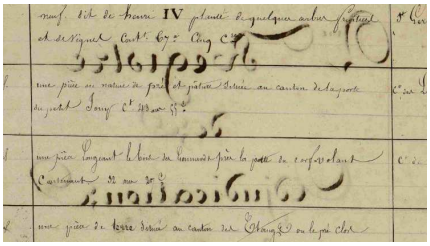
1.2.1.3 Altérations au cours du cycle de vie du document

Depuis sa création jusqu'à son analyse sous forme numérique, un document ancien a subi un grand nombre de perturbations qui vont altérer tant son contenu que sa structuration. La figure 1.3 illustre quelques dégradations classiques.

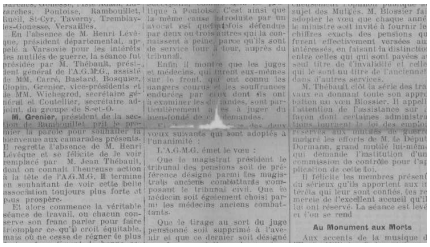
Lors de sa création, les auteurs d'un document peuvent faire des ratures, des tâches d'encre, et altérer dès l'origine le message à transmettre. Ensuite, le document va subir nombre de dégradations physiques au cours du temps, impactant l'intégrité du support et de l'encre, entraînant une perturbation de ses éléments structurants : courbure des traits, effacement de texte, traversée de l'encre entre les faces des pages, pliures, déchirures, etc. Le fait que le document puisse être annoté ou complété au cours du temps renforce ces difficultés, car l'information peut alors être étendue, altérée ou diminuée, rendant délicate la distinction entre l'information pertinente et le bruit. Finalement, d'autres défauts peuvent également survenir pendant l'étape de numérisation, comme les biais, les courbures, la rognure de zones, qui contribuent à dégrader la qualité de l'information véhiculée par le média. Le stockage sous forme numérique, avec d'éventuels recadrages, compressions et autres pertes d'informations imposées par les contraintes de stockage, finit d'altérer le document.



(a) Tache



(b) Traversée d'encre



(c) Pliures

(d) Courbure

FIGURE 1.3 – Quelques exemples de dégradations classiques dans les documents anciens numérisés.

Dans le contexte de la reconnaissance d'écriture manuscrite, Lorette [66] propose d'assimiler l'écriture à un canal de communication bruité qui nécessite des méthodes d'analyse permettant la correction d'erreurs. Ce constat reste valable à l'échelle du document et le problème inverse de récupération du message original nécessite des modèles de document tolérant le bruit et les erreurs.

1.2.1.4 Variabilités au sein d'un grand volume de documents

Le volume de données à traiter de le cas des fonds d'archives joue un rôle amplificateur dans les difficultés précédemment mentionnées. La figure 1.4 montre un exemple de variabilité que nous avons pu constater dans les contenus des documents anciens que nous avons traités, et la figure 1.5 illustre la variabilité de structure entre certains registres.

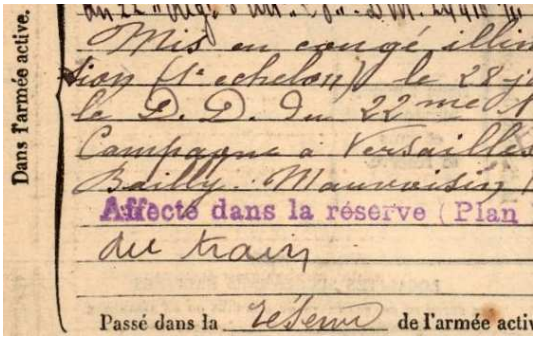
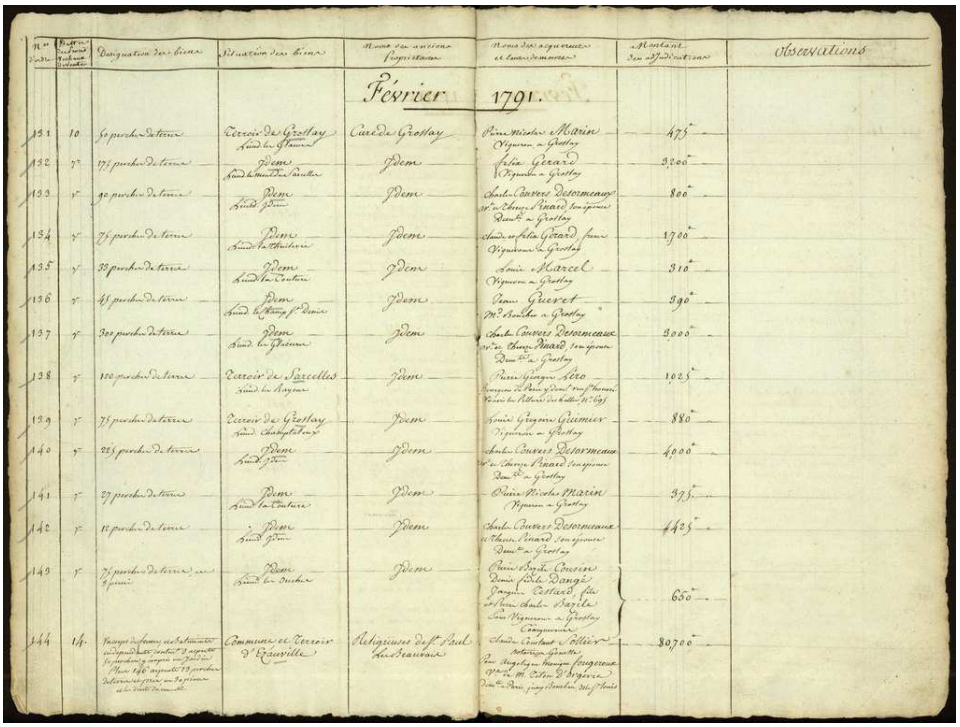
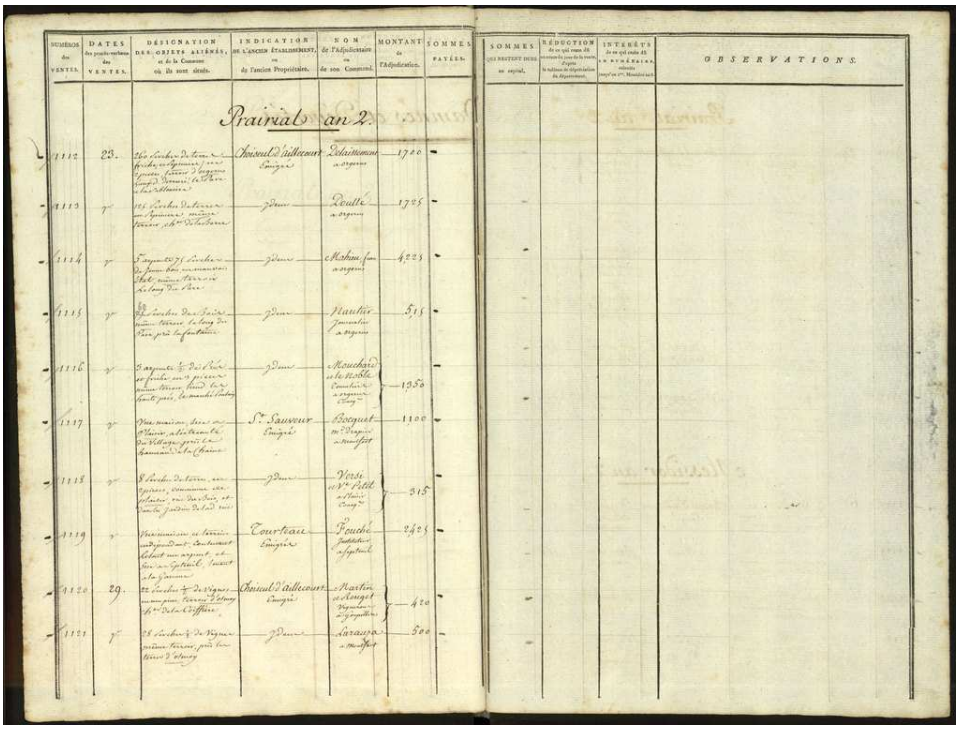


FIGURE 1.4 – Exemple de variation dans les types d'écritures.



(a) Registre à sept colonnes significatives



(b) Registre à six colonnes significatives

FIGURE 1.5 – Exemple de variation structurelle au sein d'un fonds documentaire.

Chaque document est unique, et a pu subir des modifications qui lui sont propres. Malgré l'apparente homogénéité d'un fonds d'archives classé à une cote archivistique précise, il est inévitable de constater de nombreuses variabilités. Variabilités au niveau des contenus tout d'abord, avec l'intervention de multiples auteurs ou annotateurs, mais aussi, dans le cas de l'écriture manuscrite par exemple, des variations entre les lexiques utilisés, les notations et les symboles entre certaines parties du fonds documentaire. Variabilités au niveau de la structuration des contenus également, c'est à dire dans la mise en forme de documents : il n'est pas rare de voir les conventions changer entre certaines parties du fonds. Nous avons par exemple constaté, lors du traitement de registres de ventes du XVIII^{e} siècle siècle, des variations dans le nombre de colonnes des tableaux (voir figure 1.5).

Le passage à l'échelle est donc une source inévitable d'incertitude quant à la forme que vont réellement revêtir les données. Coüasnon [30] témoigne, après avoir procédé à l'analyse de 165 000 formulaires du XIX^{e} siècle, qu'il est impossible d'anticiper les problèmes qui vont se produire lorsqu'on lance un traitement automatique.

On peut également indiquer que la taille des lexiques nécessaires peut rapidement devenir gigantesque si on traite plusieurs milliers de pages. Dans le cas de registres de recensement, la liste des patronymes peut atteindre plusieurs centaines de milliers de noms différents, au niveau national.

La connaissance relative aux documents qui doivent être traités ne peut donc pas être universelle pour l'ensemble des documents d'un fonds d'archives. Il faut s'attendre à ce que des erreurs se produisent.

1.2.2 Éléments favorables

Face à ces difficultés, il serait déraisonnable d'imaginer qu'il existe des solutions qui permettront de corriger toutes les erreurs qui se produiront. Toutefois, il est envisageable de tirer profit de certaines éléments favorables afin de guider, corriger, optimiser ou compléter l'interprétation de documents et atteindre un niveau de qualité acceptable pour la tâche considérée.

1.2.2.1 Contexte documentaire

Le contexte documentaire, que nous avons introduit en sous-section 1.1.4, permet de tirer profit de certaines propriétés intéressantes, et de *faire du volume un atout*. Que ce soit au sein d'une page ou de plusieurs, il est possible de :

- construire des systèmes de contraintes entre certains éléments, comme par exemple dans le cas des numéros de pages ;
- profiter des redondances entres les éléments « similaires ».

L'exploitation des redondance est particulièrement intéressante car elle permet de profiter de ce qu'on pourrait appeler un *principe de stabilité locale* qui s'appliquerait à différents niveaux d'abstraction du *« langage documentaire »* . Au niveau lexical, on peut alors faire l'hypothèse d'une certaine stabilité au niveau morphologique, qui permet par exemple, selon Nagy [80], de profiter de la stabilité d'une fonte dans un même mot afin de lever certaines ambiguïtés, comme le cas du *i* majuscule (`I`) et du *l* minuscule (`l`) qui seraient similaires dans deux fontes différentes, grâce aux caractères voisins.

Déjean et Meunier [32] exploitent ainsi la stabilité des titres en haut de page sur les pages voisines d'un livre, et nous avons également tiré profit de la stabilité du scripteur sur des pages voisines des registres de ventes présentés en sous-section 1.1.2.2. Dans le cas de

documents à la structure stable, comme des formulaires d'enrôlement miliaires par exemple, il est possible de profiter des images les moins dégradées pour détecter les dimensions des cases d'un type de formulaire et se servir de cette connaissance pour détecter les éléments structurants dans des images moins bien conservées.

1.2.2.2 Présence de l'humain au cours du processus

Nous reviendrons sur ce point au chapitre 4, mais nous pouvons déjà indiquer que la présence (pour ne pas dire l'omniprésence) de l'humain au cours du processus de dématérialisation des documents, et tout particulièrement à la fin, lors de la consultation, peut être un avantage. L'humain bénéficie, en effet, de facultés supérieures à la machine pour l'analyse de fonds documentaires, en particulier dans sa capacité à exploiter un contexte extrêmement vaste lors de son interprétation [82, table 1]. Ceci pose la questions du ou des rôles à accorder à l'humain pour pouvoir bénéficier de ses capacités.

Précisons ici que dans ce manuscrit, nous ne nous intéresserons pas aux aspects ergonomiques liés à la réalisation d'interfaces homme–machine, mais nous accorderons de l'importance aux données échangées et aux scénarios décrivant ces échanges.

1.2.2.3 Méthodes d'apprentissage artificiel

Nous appelons « apprentissage artificiel » la discipline qui vise à décrire les techniques permettant à un apprenant virtuel de trouver une fonction de classification ou de régression à partir d'observations du comportement d'un système idéal. Nous reprenons de Cornuéjols *et al.* [20] cette traduction du terme *« machine learning »*. Bien que l'utilisation de méthodes d'apprentissage artificiel soit tributaire d'une phase d'initialisation qui nécessite de fournir des exemples de résultats au système, les progrès en matière d'inférence de modèles structurés ou de relations spatiales, semblent permettre d'espérer une augmentation de la qualité des résultats dès qu'un nombre suffisant d'exemples a pu être généré.

Toutefois, il convient de préciser qu'il s'agit d'un tout autre domaine de recherche que celui exploré dans cette thèse, et notre principal souci dans ce travail sera d'être compatible avec ces avancées pour permettre leur intégration, principalement en favorisant l'apport d'informations utiles à un système d'apprentissage au cours de la phase délicate d'amorçage.

1.2.2.4 Méthodes de reconnaissance d'écriture manuscrite

À l'instar du domaine de l'apprentissage artificiel, le domaine de la reconnaissance de l'écriture manuscrite a largement progressé au cours de la dernière décennie. Une meilleure exploitation du contexte linguistique, ainsi qu'une meilleure tolérance aux variations du signal, ont permis la création d'outils présentant des taux de reconnaissance capables d'affronter des situations réelles [96, 121, 122].

De même que pour les techniques d'apprentissage artificiel, nous rechercherons dans ce travail à permettre l'exploitation de ces avancées, sans faire de proposition dans ce domaine.

1.3 Conclusion

Ce premier chapitre a permis d'introduire un certain nombre de notions essentielles pour la présentation de notre travail. Le *contexte documentaire* (voir sous-section 1.1.4), en particulier, sera un concept central dans le reste de la présentation de nos travaux. Rappelons

que l'exploitation de ce dernier est indispensable lors de l'interprétation fine du contenu des documents. Cette interprétation est nécessaire pour fournir à l'utilisateur final de l'environnement de consultation dématérialisé (le lecteur) la possibilité d'exploiter pleinement les ouvrages mis à sa disposition, sans trop subir ce changement d'usage.

On a vu que les documents anciens présentent de nombreuses difficultés. À ce titre, ils peuvent être considérés comme des objets scientifiques représentatifs d'une grande catégorie de documents.

Ce chapitre a également été l'occasion de préciser le contour de notre démarche scientifique, et de commencer à identifier les contraintes que doit satisfaire un système capable de transcrire des documents anciens, grâce à une interprétation des contenus. Ce problème de transcription nécessite de nombreuses précautions afin d'éviter que le processus d'extraction automatique de l'information soit globalement plus coûteux qu'une version intégralement manuelle [6].

Les prochains chapitres de cette première partie seront donc consacrés à l'étude de trois aspects essentiels de l'interprétation de fonds documentaires.

- Le chapitre 2 s'intéresse à la façon dont les approches existantes *expriment la connaissance à priori* à propos des documents et *automatisent l'utilisation* cette dernière. La disponibilité de connaissances relatives aux contenus des documents et à leur organisation est le principal élément limitant les possibilités de transcriptions de fonds d'archives.
- Le chapitre 3 détaille comment les approches existantes *représentent et exploitent automatiquement le contexte documentaire*. Considérer le fonds dans sa globalité permet l'expression de nouvelles propriétés et leur exploitation automatique. Ce chapitre permettra de d'identifier des éléments indispensables pour fiabiliser le travail du système, mais aussi pour le guider, le valider, ou le compléter ; c'est à dire pour faire du volume de documents un atout.
- Le chapitre 4 se focalise sur les manières dont les approches existantes *tirent profit de la présence de l'humain et des capacités de ce dernier*. La présence de l'humain tout au long du processus de dématérialisation des documents, à des moments et avec des rôles multiples, doit être mieux comprise et utilisée, afin de permettre de tirer profit des capacités supérieures de l'humain à interpréter des fonds documentaires.

Le chapitre 2 permettra de dessiner le contour des outils de référence que nous proposerons ensuite d'étendre, et les chapitres 3 et 4 permettront de mettre en lumière les éléments à intégrer dans le nouveau système d'interprétation de fonds documentaires que nous décrirons dans la partie II.

Chapitre 2

Interprétation de documents image par image

Introduction

Nous nous focalisons à présent sur l'étape d'*extraction de contenus* du processus de dématérialisation, qui consiste à passer d'une représentation de l'information sous forme d'image isolée (nous la considérerons en contexte avec d'autres pages dans le chapitre 3) à une représentation informatique structurée permettant une exploitation sémantique de son contenu. Dans le cas des registres de ventes présentés en introduction, dont un extrait est visible en figure 2.1, il pourrait alors s'agir de produire les entrées illustrées en table 2.1 dans une base de données.

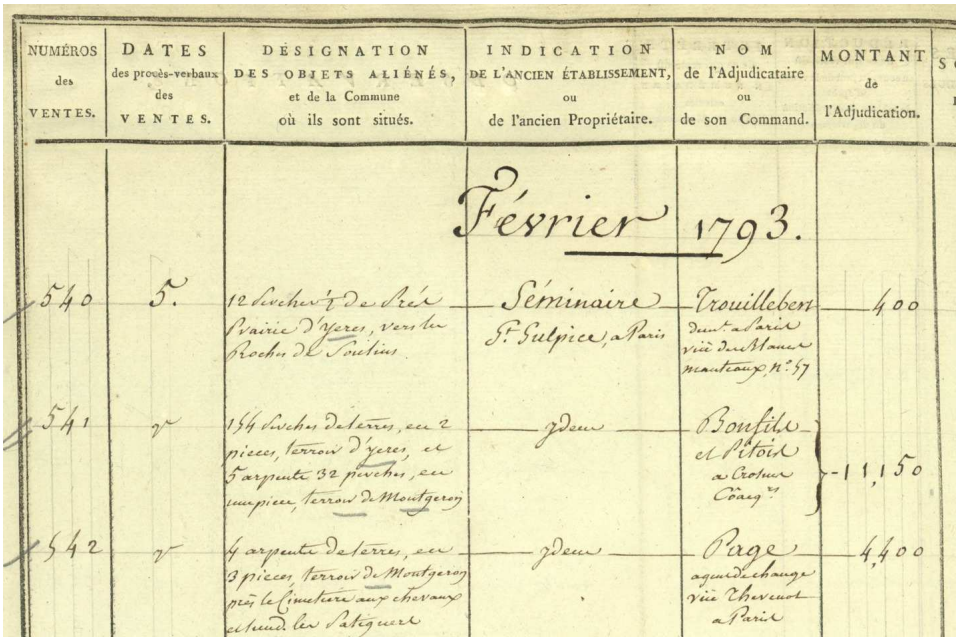


FIGURE 2.1 – Extrait d'une image de registre de ventre du XVIII^{e} siècle.

Numéro	Date	Ancien propriétaire	Acquéreur	...
540	5 février 1793	Séminaire St Sulpice	Trouillebert	...
541	5 février 1793	**Séminaire St Sulpice**	Bonfils et Pitois	...
542	5 février 1793	**Séminaire St Sulpice**	Page	...
...	...	...	...	...

TABLE 2.1 – Exemple de données qu'il est possible d'extraire de l'image figure 2.1. Les entrées notées en **gras** ont nécessité une interprétation sémantique de l'image afin de retrouver le sens des entrées où *« idem »* est indiqué.

Pourquoi nous intéressons-nous à un mode d'interprétation image par image ? Pour deux raisons : tout d'abord parce que l'information pertinente est majoritairement contenue dans l'image, même si elle peut être ambiguë ou fragmentaire ; et ensuite parce que ce mode de travail correspond à l'immense majorité des systèmes existants. Par conséquent, limiter l'effort d'adaptation d'un système existant, ainsi que l'effort de conception, requis pour mettre en place un système capable de traiter de façon innovante des fonds documentaires nécessite de respecter ce paradigme et le compléter au besoin.

Nous allons donc nous intéresser dans ce chapitre à la représentation des connaissances à propos des contenus (et de leur organisation) des images et à la façon dont ces dernières sont utilisées au cours du processus d'interprétation. Nous nous intéresserons également à l'acquisition de ces connaissances, et à la façon dont elles permettent de tolérer les difficultés mentionnées en sous-section 1.2.1 : ambiguïtés liées au type de document, ou dégradations que ce dernier a subies. Par la suite, nous nous efforcerons de distinguer au moins deux étapes lors de la construction d'un système d'interprétation automatique :

- l'étape de *conception* où le système est généré ;
- et l'étape d'*exploitation* ou d'*exécution* où le système est utilisé pour traiter des images.

Nous proposons ici d'identifier les éléments pertinents pour le problème d'interprétation en regroupant les approches représentatives selon un plan proposé par Pasternak [92] qui distingue d'un côté les méthodes basées sur des algorithmes, et de l'autres celles qui spécifient de façon déclarative (partiellement ou non) les connaissances relatives aux documents traités. Nous avons ajouté une section relative aux approches basées sur des modèles statistiques, qui tendent selon nous à se situer à mi-chemin entre ces deux approches, dans le sens où la connaissance sur laquelle elles se basent reste implicite, mais se sépare de l'algorithme d'analyse.

Afin de ne pas limiter notre étude, nous avons considéré les travaux sur l'interprétation de documents variés, tout en conservant nos contraintes liées au traitement de fonds d'archives. Pour terminer cette introduction, nous listons rapidement les principaux types de documents traités par les approches que nous considérons dans ce chapitre, et évaluons leur ressemblance avec les documents historiques qui nous intéressent.

Documents majoritairement textuels Cette première catégorie est celle qui a reçu le plus d'attention de la communauté scientifique. Elle est composée des chèques bancaires, des enveloppes de courrier, des formulaires, des factures, des documents administratifs en général, des courriers, de la presse... Dans le cas des chèques, des adresses postales et des formulaires, l'état de l'art actuel offre des performances permettant une utilisation intensive en production [42]. Tous ces types documents présentent beaucoup de similarités avec les documents d'archives qui peuvent parfois être de la

même nature (presse et courrier en particulier).

Documents majoritairement graphiques Sous cette catégorie, on peut regrouper les plans d'architecture, les schémas électiques ou hydrauliques, les cartes géographiques et plans cadastraux, ainsi que les dessins techniques. Ces types de documents nécessitent en général des stratégies d'analyse spécifiques, mais présentent des difficultés similaires à celles présentées en section 1.2.1.

Documents « nés électroniques » Sous cette catégorie (non disjointe des deux précédentes), nous regroupons les documents de type PDF, Postscript, XML et autres formats permettant une identification immédiates des contenus du document. Dans ces cas, les méthodes d'analyse n'ont pas besoin de détecter de primitives visuelles, et peuvent se concentrer sur une transformation de format. Bien que certaines ambiguïtés puissent persister, cette tâche de transformation est en général beaucoup plus aisée que celle d'interprétation d'images.

2.1 Connaissance sous forme algorithmique

La communauté scientifique de traitement de document a été fortement influencée par la communauté de la reconnaissance de formes *(Pattern Recognition)* d'où elle a émergé. La problématique de l'analyse du document s'est construite progressivement, en s'intéressant d'abord à l'identification d'éléments isolés, comme le font remarquer Nagy et Veeramachaneni [82] : qu'il s'agisse de glyphes, d'illustrations, ou de pages complètes, les travaux fondateurs se sont efforcés de proposer des solutions permettant d'associer à un fragment de signal (un tableau de pixels) une *classe*, au sens de la reconnaissance de formes, c'est à dire une interprétation plus au moins abstraite de son contenu.

Afin de permettre d'appliquer les techniques de reconnaissance de caractères imprimés (OCR), de caractères manuscrits, ou de symboles, les premiers travaux se sont donc naturellement tournés vers l'isolation des formes qu'il fallait identifier (symboles de composants électroniques, lignes de texte, etc.) dans les images représentant généralement une page du document à interpréter.

2.1.1 Approches par regroupements et par divisions

O'Gorman et Kasturi [89] donnent en 1995 une vision de l'approche standard pour les tâches d'OCR et de reconnaissance de symboles, où après une phase de filtrage visant à simplifier le signal de l'image (suppression de divers bruits, binarisation, voire vectorisation), il devient possible de commencer à détecter l'organisation spatiale des éléments de l'image. Pour les documents textuels, il s'agit alors de localiser les lignes de texte avant de les transmettre à un module d'OCR. Dans le cas des documents imprimés aux structures simples, ces techniques continuent de faire leurs preuves. Partant de cette représentation quasi-brute de l'image, on distingue classiquement deux types d'approches pour isoler les symboles à reconnaître. [1]

1. Nous faisons ici une présentation très générale de ces approches car elles sont déjà décrites de façon exhaustive ([114, 115, 81, 83]) et nous ne nous intéresserons pas dans la suite aux traitements au niveau signal.

2.1.1.1 Travail au niveau du signal

La première famille procède par regroupements [2] successifs de pixels voisins pour former des mots, lignes ou paragraphes de texte à reconnaître. Ces approches sont particulièrement efficaces pour construire de nouvelles primitives graphiques, comme des composantes connexes ou des lignes, par exemples, qui pourront ensuite être exploitées par des niveaux d'analyse plus abstraits.

La seconde famille procède par divisions successives de l'image à analyse jusqu'à atteindre les symboles à identifier. Ces approches sont encore largement utilisées pour découper grossièrement les images de documents et disposer d'une première estimation de l'organisation spatiale des principaux cadres.

La principale raison pour laquelle ces approches ne peuvent que représenter une étape préliminaire de l'interprétation d'image de document est bien résumée par Lladós *et al.* [65] qui indiquaient en 2002 que la *segmentation* des images de documents (c'est à dire *l'isolation spatiale des formes à reconnaître*) était un des problèmes non résolus dans le cas de la reconnaissance de symboles. En effet, de nombreuses approches se basent sur l'hypothèse que les symboles à reconnaître ont déjà été localisés. Cependant, comment le précisent les auteurs, le fait de pouvoir dissocier deux étapes dans le problème de reconnaissance de symboles dans des images, à savoir : (i) la segmentation, et (ii) la classification, était caractéristique de documents pour lesquels il est possible après quelques pré-traitements d'isoler les éléments pertinents avec un traitement local et hors contexte, comme la détection de composantes connexes. Pour des documents à la structure plus complexe il est nécessaire de s'appuyer sur une connaissance spécifique, voire de faire appel à un humain. Ceci fait écho au paradoxe *« il faut segmenter pour reconnaître, et reconnaître pour segmenter »* qu'on attribue à Sayre [107], et montre bien l'importance d'une interprétation contextuelle intégrée à la localisation des symboles ou des zones d'intérêt d'une image de document.

2.1.1.2 Quelques définitions

Dans la mesure où la reconnaissance de caractères imprimés (OCR) a été la finalité de nombreux travaux dans le domaine du document, il n'est pas surprenant que le vocabulaire employé fasse implicitement référence au processus « *segmentation* $\rightarrow$ *reconnaissance* » de caractères supposés bien segmentés.

Selon Tang *et al.* [114, 115], on peut distinguer deux phases (hors pré-traitements) lors du traitement d'une image de document (textuel) :

1. la phase d'*analyse (analysis)* du document, qui consiste à trouver la décomposition hiérarchique en blocs la plus vraisemblable selon un critère donné : on obtient alors une structure dite « géométrique », ou aussi « physique », décrivant la mise en page *(layout)* ;
2. la phase de *compréhension (understanding)* du document, qui consiste à passer de cette représentation « géométrique » à une autre représentation, dite « logique », et décrivant les entités sémantiques du documents (titre, illustration, article, etc.) et les liens (ordre de lecture, relation entre le titre d'un article et le corps de celui-ci, etc.) entre ces dernières.

2. Ces approches sont souvent appelées « ascendantes » quand elles procèdent par regroupements, et « descendantes » quand elles procèdent par divisions. Toutefois, ces termes sont parfois utilisés avec un sens différent pour désigner des techniques d'analyse « guidées par les données », ou « guidées par un but » ([123]). Nous n'utiliserons donc pas ce vocabulaire.

Il est courant d'ajouter à ces deux étapes une troisième dite de *« reconnaissance »* du texte imprimé qui permet d'associer une valeur (textuelle ou numérique) aux fragments isolés d'images de symboles, de mots ou de lignes grâce à un système de classification.

Ce schéma d'analyse est résumé dans la figure 2.2 extraite de [114] illustrant le processus d'interprétation d'une page de presse.

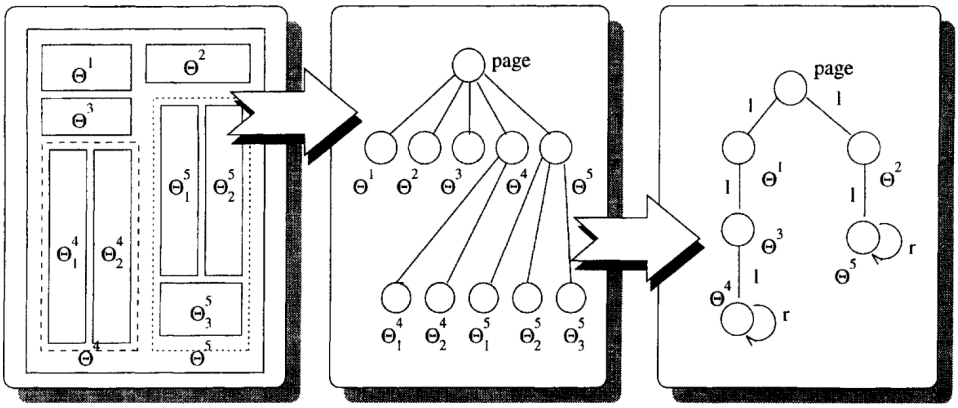


FIGURE 2.2 – Analyse et compréhension de documents, selon Tang *et al.* (Reproduit d'après [114, figure 2].)

Dans cette thèse, le terme d'*interprétation* que nous employons se veut plus général que le terme *understanding* proposé par Tang *et al.* , afin de ne pas pas nous limiter à un modèle imposant ce découpage en deux temps. Toutefois, l'objectif de production d'une structure de niveau sémantique reste notre objectif, et nous ne nous préoccuperons pas des traitements réalisés au niveau du signal de l'image. Nous adoptons, en ce sens, la définition large proposée par Nagy [81] :

> « La *compréhension d'images de documents* (ou *interprétation*) est la construction d'une représentation formelle des relations abstraites signifiées par l'arrangement bi-dimensionnel des symboles. »

On peut donc dire que notre travail traite de l'interprétation symbolique d'images de documents.

2.1.2 Approches hybrides

Shafait *et al.* [112, 111] ont montré que prises séparément, les meilleures des méthodes (par regroupements ou par divisions) présentées en sous-section 2.1.1.1 présentent des performances très différentes selon les types de documents traités et leur niveau de dégradation.

Bien avant ces résultats, de nombreux systèmes ont tenté de tirer profit des avantages et inconvénients des méthodes de segmentation en les combinant pour former des approches *hybrides*, en créant une combinaison de transformations plus performante sur un type de document donné. On voit alors apparaître une *connaissance* relative au document traité dans les systèmes proposés.

2.1.2.1 Systèmes OCR prototypes

Des exemples d'approches hybrides récentes sont les environnement de développement de prototypes de systèmes OCR dédiés à un type de document tels que OCROpus, déve-

loppé par Bruel [14], et Gamera, développé par Droetboom *et al.* [34, 35]. Ces systèmes proposent un environnement logiciel permettant d'enchaîner facilement des traitements au niveau de l'image afin de localiser des lignes de texte (imprimées) et reconnaître leur contenu.

Le manque de structures d'abstraction de ces outils rend toutefois fastidieuse la gestion de mises en pages complexes, ainsi que la tolérance aux différentes formes de dégradations classiques. Dans ce type de système, la connaissance relative à la structure du document analysé est donc programmée par un expert, et cette connaissance reste implicite dans l'enchaînement des conditions d'application des transformations à l'image. Le domaine d'application privilégié de ce type d'approche semble donc être l'analyse superficielle des documents imprimés de relativement bonne qualité.

2.1.2.2 Outils d'enchaînement de traitements niveau image

L'enchaînement de traitements (binarisation, classification d'une zone en un type « texte » ou « graphique », appel d'un module OCR, etc.) nécessite une grande cohérence entre les interfaces des briques logicielles utilisées. C'est une source d'erreur lors de la conception de ces outils. Le projet DocMining s'attaque à ce problème. DocMining [1, 19] est un outil de construction de systèmes d'analyse de documents, qui facilite l'assemblage de nombreux composants pour réaliser une séquence de traitements.

Cet outil permet de spécifier, à l'instar des logiciels de conception de composants électroniques, une composition d'unités de traitement d'information *(Data Processing Units)* spécialisées dans une tâche particulière, acceptant un certain type de données et en produisant un autre. Comme dans le cas d'OCROpus, les interfaces sont spécifiées précisément. L'enchaînement des opérations à réaliser à partir de l'image brute est spécifié par un expert et forme un *« scénario »*. Afin de garantir la cohérence d'un scénario, l'outil permet de contrôler la compatibilité entre les différentes entrées et sorties des unités de traitement. Finalement, pour faciliter l'échange d'informations entre les différentes unités de traitement, ces dernières travaillent exclusivement sur une même structure globale représentant une vue abstraite du document sous la forme d'un arbre DOM [3].

La figure 2.3 (extraite de [19]), illustre un scénario d'OCR basé sur les étapes suivantes :

1. Binariser l'image.
2. Réaliser une séparation entre contenus textuels et graphiques.
3. Réaliser une segmentation en blocs de texte pour la couche textuelle.
4. Réaliser une segmentation en composantes connexes pour la couche graphique.
5. Réaliser une segmentation en composantes connexes pour les parties de l'image qui n'ont été détectées ni comme textuelles, ni comme graphiques.
6. Corriger les erreurs de séparation texte — graphique en comparant les résultats des trois segmentations selon une technique définie par le concepteur.
7. Réaliser l'OCR sur les blocs de texte de la couche textuelle.
8. Réaliser la vectorisation des éléments de la couche graphique.

Dans ce type d'approche, la mise au point d'un système performant pour le traitement de documents anciens peut s'avérer fastidieuse pour plusieurs raisons. L'enchaînement linéaire des traitements, tout d'abord, rend les premières étapes critiques pour les suivantes,

3. DOM *(Document Object Model)* : Standard de représentation abstraite de documents proposé par le *World Wide Web Consortium* (voir `http://www.w3.org/DOM/`).

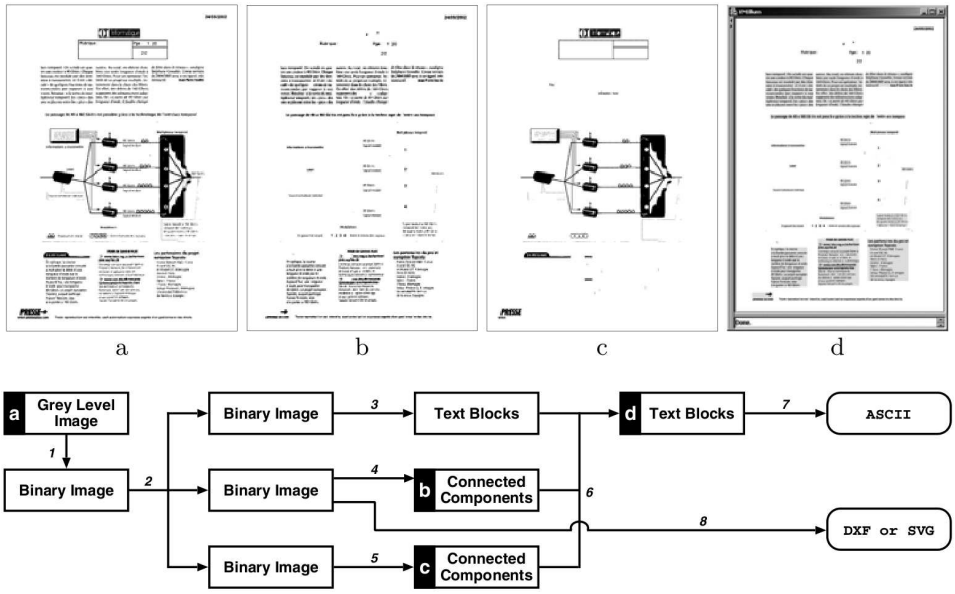


FIGURE 2.3 – Exemple de scénario d'OCR pour le système DocMining. (Reproduit d'après [19, figure 3])

afin d'éviter des *cascades d'erreurs* : une erreur dans l'étape n produit k erreurs dans l'étape $n+1$, et ainsi de suite... Les documents anciens ou dégradés sont alors difficiles à analyser à cause des erreurs inévitables. Ensuite, on peut imaginer que la fragmentation dans de multiples procédures des connaissances relatives au type de document interprété, ainsi que la mise au point par cycles « *essai* $\rightarrow$ *analyse des erreurs* $\rightarrow$ *corrections* » rendent lourde et peu fiable l'adaptation du système à des documents complexes. Finalement, on peut remarquer que la faible structuration des résultats produits ne semble permettre qu'une interprétation superficielle du contenu du document.

2.1.3 Nécessité d'un modèle de document

En conclusion de cette section, on peut indiquer que les approches purement algorithmiques permettent d'isoler simplement des primitives visuelles (composantes connexes, blocs de texte, etc.) lorsqu'aucune interprétation du contenu n'est nécessaire, et que la décision locale ne dépend pas d'un contexte plus large, spécifique au type de document concerné. Les documents spécifiquement conçus pour une segmentation hors-contexte (formulaires modernes, etc.) pourront être traités de cette façon, mais les documents anciens et autres types de documents où la composition est relativement libre poseront problème. On pourra se référer à [123] pour une classification des types documents selon leur structuration et le niveau de contexte nécessaire à l'interprétation de leurs contenus.

L'analyse de documents complexes et dégradés est donc plus délicate. Mao *et al.* [72] résument bien les limitations des approches algorithmiques : selon ces auteurs, les principales limites des méthodes d'analyse de structure d'images de documents sont :

- de ne pas être basées sur des modèles formels ;
- de se baser généralement sur l'hypothèse selon laquelle l'isolation spatiale des principaux objets à identifier a déjà été réalisée ;

– d'échouer à produire des résultats corrects en présence de bruit ou d'ambiguïtés.

Pour faire face à ces limites, il est donc nécessaire de baser la conception d'un système d'interprétation (qui soit capable de faire face à des documents à structure complexe) sur un modèle de document, c'est à dire *une description de l'ensemble des éléments graphiques qu'il peut comporter, la façon dont ces derniers peuvent être composés, et les liens entre eux (leur sémantique)*, afin de sélectionner automatiquement, lors de l'exécution du système, les algorithmes les plus adaptés pour extraire l'information utile.

2.2 Connaissance sous forme statistique

Les approches basées sur une représentation statistique de la connaissance bénéficient, à priori, de deux avantages majeurs :

1. il est possible de générer cette connaissance de façon semi-automatique grâce à des exemples, ce qui peut faciliter la phase de conception ;
2. des méthodes éprouvées permettent un appariement entre un modèle statistique et des données observées afin de reconnaître les informations utiles, et permettent alors de baser la phase d'exploitation des connaissances, lors de l'exécution du système, sur un socle générique (réutilisable) et performant.

Cette section s'intéresse brièvement aux approches basées sur cette forme de connaissance, et sur leurs limites.

2.2.1 Modèles graphiques probabilistes pour l'exploitation d'un contexte local

Les modèles graphiques probabilistes sont des modèles statistiques qui utilisent des graphes pour représenter les relations entre des variables. Ceci permet de modéliser, dans une certaine mesure, des relations contextuelles entre des objets.

À notre connaissance, ces approches peuvent soit être basées sur des réseaux bayésiens [58], soit s'appuyer sur des champs aléatoires de Markov [62, 79]. Dans les deux cas, il s'agit d'associer à une zone de l'image (qui peut être un pixel) une étiquette logique en considérant les caractéristiques du voisinage de cette zone.

Les caractéristiques utilisées peuvent être assez variées :
– police, taille, couleur de caractères isolés ;
– localisation dans l'image (positionnement par rapport au bord haut, bas, etc.) ;
– présence de mot-clé ;
– niveau moyen de luminosité dans différentes directions du voisinage...

Chaque ensemble de caractéristique est encodé dans un vecteur de taille fixe décrivant chaque portion de signal considérée.

2.2.2 Faible structuration et acquisition difficile des connaissances

Ces approches montrent des performances raisonnables dans les récentes compétitions sur les courriers manuscrits grâce à leur capacité à tolérer les dégradations et l'ambiguïté. Toutefois, la difficulté à intégrer de nouvelles informations au codage à cause de la représentation vectorielle de l'information, ainsi que la difficulté à représenter l'organisation des contenus entre eux, en particulier de façon hiérarchique, limitent ces approches à une détec-

tion, au niveau signal, de classes qui peuvent être déduites des informations locales. Pour ces raisons, ces approches sont principalement utilisée pour des données peu structurées.

Par ailleurs, on peut noter l'existence sous forme implicite d'une connaissance à priori dans le choix des caractéristiques des observations et des relations entre ces dernières. Cette connaissance n'évolue pas au cours des apprentissages statistiques réalisés pour entraîner le système.

Finalement, l'apprentissage de ces systèmes nécessite une quantité non négligeable de données correctement étiquetées, ce qui est rare et coûteux dans le cas des documents anciens. Cantonner cette phase d'apprentissage en amont de la phase d'exploitation semble donc problématique.

2.3 Connaissance sous forme déclarative

Nous nous intéressons à présent à une autre catégorie d'approches qui visent à expliciter la connaissance à priori à propos des documents qu'elles permettent de traiter. Ces approches permettent à un expert de décrire le document à traiter lors d'une phase de conception et s'appuient ensuite sur cette description pour extraire les données utiles.

La force de ces approches est de tirer profit des spécificités des documents considérés et de permettre une meilleur gestion des dégradations, ou encore la vérification de contraintes entre les éléments reconnus : par exemple, l'égalité entre la somme des nombres d'une colonne « prix » d'un tableau et la case « total ».

Un autre avantage majeur de la séparation entre le modèle de document et l'algorithme d'appariement est la capacité du système à être facilement transposé à un autre problème en modifiant en totalité ou en partie le modèle. Dans ce cas, on pourra parler d'approches *génériques*, par opposition aux approches *spécifiques* à un type de document.

Parmi ces approches, la formalisation de la connaissance peut être plus ou moins explicite, et plus ou moins dissociée du système chargé de réaliser l'appariement entre les données analysées et le modèle. Nous avons donc distingué trois types d'approches :

1. les approches semi-déclaratives, pour lesquelles une partie de la connaissance reste sous forme algorithmique, et donc implicite ;
2. les approches purement déclaratives, qui se basent sur des modèles stricts des documents à analyser ;
3. les approches déclaratives extensibles, qui permettent des descriptions flexibles exprimant facilement de nouvelles propriétés à propos des documents.

2.3.1 Approches semi-déclaratives

Un première façon d'exprimer des connaissances à propos du contenu à détecter dans des images de documents est de décrire sous forme de règles les relations entre les éléments abstraits qu'on cherche à produire et les éléments qui composent la forme structurée en question, ainsi que les contraintes associées à ces éléments (relatives et intrinsèques). Par exemple, dans le cas d'un courrier, on pourrait formuler la règle suivante : « un bloc de description du destinataire doit être placé en haut de page, et formé d'au moins trois ligne verticales alignées à gauche, la première contenant un nom, et la dernière un code postal. »

Vaxivière et Tombre [119] ont procédé de cette manière pour mettre en œuvre un prototype de système d'interprétation de dessins industriels mécaniques. Leur objectif était de

retrouver la structure de schémas pour permettre l'exploitation de ces derniers avec un logiciel de conception assistée par ordinateur. Une des tâches consistait alors à identifier les blocs de matière dans un dessin technique. Voici un exemple de règle intégrée à ce système : « Tout bloc rectangulaire ou trapézoïdal, symétrique par rapport à la ligne pointillée de référence, et voisin d'un bloc déjà reconnu comme bloc de matière, est aussi un bloc de matière. »

En conclusion de leur travail sur de ce prototype, ces auteurs indiquent que leur approche a exploité un ensemble de règles sans structuration particulière, et que cette représentation « à plat » de la connaissance était difficile à faire évoluer. Ils suggèrent alors d'organiser davantage l'information, en suggérant une *représentation hiérarchique*, éventuellement à l'aide d'un *langage dédié*.

Dans le cadre de l'interprétation d'image des formulaires et de journaux, Watanabe *et al.* [123] ont également utilisé des ensembles de règles faiblement structurées, en précisant la nécessité d'intégrer un mécanisme de retour-arrière *(backtrack)* pour permettre de gérer les ambiguïtés d'interprétations des différentes entités visuelles.

Le principal reproche qu'on peut faire à ces approches est leur spécificité qui rend difficile leur application à d'autres types de documents. On peut également indiquer que la forte dépendance des ces approches à l'étape préalable de segmentation les rend très sensibles aux ambiguïtés et incertitudes des documents anciens. Finalement, la faible structuration de la connaissance limite la capacité de modélisation du document et ne permet qu'au prix de lourds efforts (consistant principalement à vérifier la cohérence des règles entre elles) de produire une structure résultat complexe.

2.3.2 Approches purement déclaratives

Afin de s'affranchir des problèmes d'adaptation et de maintenance des systèmes basés sur des connaissances à priori, certains travaux ont cherché à séparer cette connaissance du mécanisme d'analyse, afin de permettre :

- une génération automatique d'un système spécifique à un type de document, grâce à des outils de description et d'appariement génériques ;
- une plus grande flexibilité dans le mécanisme d'exploration de l'espace des hypothèses d'interprétation possibles (et donc une meilleure gestion de l'ambiguïté) grâce à une indépendance du mécanisme d'appariement entre le modèle de document et l'image. Ce mécanisme peut intégrer des heuristiques, et être amélioré sans modifier la description factuelle du contenu de l'image.

2.3.2.1 Premiers langages de descriptions

Une des premières approches complètes dans cette direction a été celle de Tang *et al.* [115, 113, 114] destinée au traitement automatique de chèques bancaires et de formulaires. Cette approche est basée sur un langage de description de formulaires *(FDL)* qui permet de décrire les deux types d'éléments indiqués ci-après.

1. Les éléments *structurants* de l'image, à savoir des lignes et des bandes. Pour chaque élément attendu, il est possible de spécifier s'il s'agit d'une ligne ou d'une bande, l'orientation (verticale, horizontale, travers), l'épaisseur attendue (fine, moyenne, large) et la position attendue (coordonnées du point haut gauche et longueur).
2. Les éléments *à localiser et reconnaître* dans l'image, à savoir des fragments de texte imprimés ou manuscrits. Pour chaque élément, il est possible de préciser son position-

nement relatif (au dessus, en dessous, à gauche et à droite) par rapport aux éléments structurants. Ceci forme un ensemble de contraintes géométriques pour la détection des contenus. Il est également possible de sélectionner un sous ensemble de la zone ainsi décrite.

L'approche de Tang *et al.* décrit le processus de création et d'utilisation d'un système de reconnaissance de la façon suivante.

1. Lors de la phase de conception, un expert décrit le document grâce au langage proposé.
2. Lors de l'exécution du système, la description est chargée par le moteur d'analyse pour permettre la recherche des éléments spécifiques au type de document traité. Grâce à des mécanismes de gestion des lignes effacées, et de résolution des contraintes de placement, le système extrait les éléments indiqués par la description.
3. Après cette phase d'exécution, les auteurs indiquent qu'il est possible de réviser automatiquement les paramètres de la description (taille et position des éléments structurants, en particulier), en fonction des variations constatées dans les documents traités.

Bien que cette approche se limite à la localisation sans structuration particulière de zones dans des documents présentant une structure très stable, à plat, et sans alternatives, elle présente toutefois un certain nombre d'avantages :

- l'utilisation d'éléments structurants pour localiser les éléments à reconnaître permet de gérer automatiquement les décalages et rotations produites par la numérisation, autorisant ainsi une certaine variabilité dans les dimensions et positions des objets ;
- la description du positionnement attendu des objets limite largement l'espace de recherche (qui peut augmenter exponentiellement selon le nombre d'éléments parasites, comme le montrent Mahoney et Fromherz [69]) et permet également de lancer un système de classification approprié pour chaque zone (en restreignant les symboles et mots possibles, dans le cas des chèques en particulier) ;
- la gestion de l'effacement des lignes peut alors être automatique, et n'alourdit pas la description ;
- une certaine adaptation des paramètres du système est possible à l'usage, dès qu'on dispose d'assez de contraintes sémantiques pour garantir qu'une interprétation considérée comme valide ne peut pas être le fruit du hasard.

2.3.2.2 Génération automatique de *blackboard*

Afin de permettre la production d'interprétations structurées, certains auteurs ont cherché à concevoir des systèmes s'appuyant sur une meilleure organisation de la connaissance et des résultats proposés.

Le modèle de *blackboard* est une architecture générique qui a été utilisée par de nombreux auteurs à cette fin. Nii [84] indique que ce modèle (qui peut être adapté à l'extrême) vise à construire progressivement une solution structurée (hiérarchiquement) à un problème. Ceci permet la mise en place de raisonnements complexes guidés par les données (dits « en avant ») ou guidés par un but (dit « arrière ») ainsi que d'autres formes plus sophistiquées. Ce modèle est basé sur trois types d'entités qui forment un système en *blackboard*.

Les sources de connaissances Parfois appelées « experts » ou « agents », elles contiennent la connaissance nécessaire à la résolution d'aspects indépendants du problème : segmentation, validation lexicale, etc.

La structure de données en *blackboard* Elle est la base de donnée servant à stocker les informations relatives à l'état courant de la résolution du problème. Les sources de données modifient progressivement cette structure pour résoudre le problème de façon incrémentale. Cette structure est le seul moyen de communication entre les sources de données.

Le contrôleur Il implémente, quant à lui, le protocole de communication entre les sources de données. La stratégie la plus simple consiste à activer la première des sources qui indique qu'elle peut apporter une nouvelle information.

Un des principaux risques de l'utilisation d'une architecture de type *blackboard* pour le traitement de documents est de la construire directement sous forme de modules séparés qui fragmentent alors une connaissance sous forme algorithmique.

Dans le cadre de la reconnaissance de dessins techniques, Pasternak *et al.* [92, 91, 93, 94] ont proposé une approche générique originale (WIZ) qui permet d'éviter de spécifier la connaissance de façon algorithmique. Les auteurs utilisent un environnement logiciel spécifique (un *framework* LISP) qui permet de générer automatiquement un système avec une architecture de *blackboard* à partir d'une description purement déclarative des contenus à identifier fournie par un expert.

Le langage utilisé permet de raisonner à partir d'objets graphiques présents dans l'image (lignes, arcs de cercles, texte, et autres éléments définis au besoin) pour former progressivement des entités de plus en plus abstraites. Il permet de définir des règles permettant de construire des objets abstraits (comme un triangle) lorsqu'un certain nombre de contraintes sont respectées (taille et position relative des traits, par exemple). Ces règles sont utilisées pour générer automatiquement des agents capables de transformer le contenu de la structure analysée, tolérant automatiquement certaines variations grâce à des mécanismes génériques.

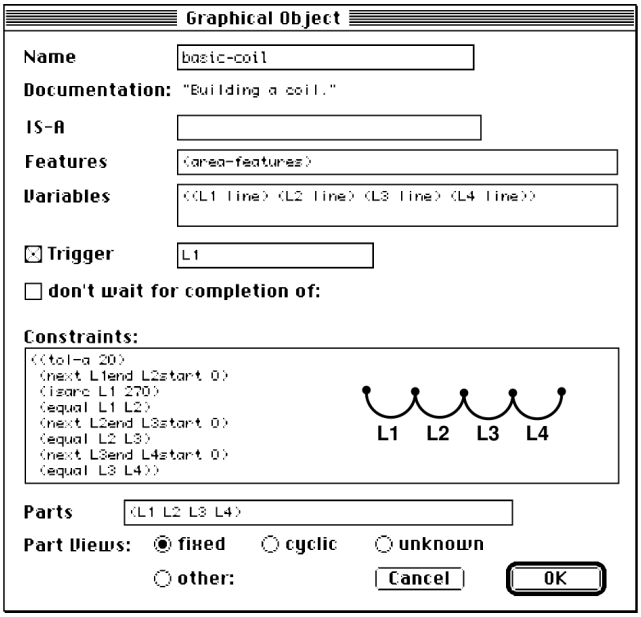


FIGURE 2.4 – Exemple de description d'un objet graphique pour WIZ. (Reproduit d'après [94, figure 6].)

La figure 2.4 montre comment cette connaissance peut être saisie par un expert et ex-

primée pour décrire la construction d'un objet abstrait (une bobine inductive) à partir d'éléments plus simple (lignes) dans un schéma électrique.

Parmi les avantages de cette approche, on peut citer la séparation claire entre la spécification des connaissances, et le contrôle de l'exploration de l'espace des solutions potentielles (ou *« espace des hypothèses »*) qui permet une certaine flexibilité dans les utilisations du système et facilite le développement.

Cependant, ce type d'approche possède un certain nombre de limitations qui font qu'elle ne semble pas adaptée à l'analyse de documents anciens. La gestion des ambiguïtés, tout d'abord, n'est faite qu'au niveau très local d'un symbole, en considérant différentes options de segmentation. Par ailleurs, le dispositif de restauration des lignes dégradées est un processus global, qui ne permet donc pas guider la recherche d'éléments structurants grâce à un modèle de document. De plus, la réduction d'objets (trois traits en un triangle, par exemple) est irréversible et ne peut être remise en cause ultérieurement, car aucun mécanisme de retour arrière n'est disponible. Finalement, le résultat final de l'interprétation est assez peu structuré, bien que l'organisation de la connaissance le soit.

2.3.2.3 Compilation de compilateurs de langages visuels

La théorie de la compilation donne un cadre théorique qui permet l'expression simple d'une description cohérente par un expert, et la génération automatique d'un système complexe d'interprétation à partir de cette dernière. La théorie des langages visuels s'intéresse à l'extension et à l'adaptation des travaux classiques de compilation (sur des espaces à une dimension) en vue d'analyser des structures bi-dimensionnelles.

Une communauté scientifique entière s'intéresse à ces problèmes, pour lesquels un excellent état des lieux est disponible dans l'ouvrage *Visual Language Theory* supervisé par Marriott et Meyer [74]. Bien que ces approches soient rarement appliquées au problème d'interprétation de documents anciens, et semblent plutôt être destinées à la construction d'outils d'édition graphique (plans, diagrammes, etc. [68]), nous pensons qu'il est pertinent de présenter rapidement les points essentiels et les avantages de ces approches.

Dans l'ouvrage précédemment cité, Marriott *et al.* [76] définissent la notion de *« langage visuel »* comme

> « un ensemble de diagrammes qui sont les « phrases » de ce langage, où un diagramme est une collection de « symboles » organisés dans un espace à deux ou trois dimensions. »

Cette catégorie d'approches s'intéresse donc à la construction de raisonnements symboliques à propos d'objets visuels.

Les auteurs précédents distinguent trois courants, non disjoints, dans les travaux théoriques :

Les approches grammaticales Elles se focalisent sur les aspects descriptifs, en décrivant le protocole de génération (ou de lecture) d'un ensemble d'objets visuels organisés selon un langage.

Les approches logiques Elles s'intéressent principalement à l'expression et à la vérification de contraintes (spatiales, numériques, etc.) entre les objets visuels.

Les approches algébriques Elles mettent l'accent sur le contrôle de validité de la composition de constructions visuelles vues comme des fonctions mathématiques, en particulier grâce à des outils comme le contrôle de typage.

Dans ce contexte, les approches grammaticales semblent avoir un rôle fédérateur car elle permettent d'intégrer des contraintes sémantiques sous forme logique, et le mécanisme de dérivation garantit dans une certaine mesure la cohérence des compositions d'objets visuels.

Une grammaire hors contexte classique [4] peut être définie de la façon suivante [5] :

$$G = \{\Sigma, N, S, P\}$$

où Σ est un ensemble de symboles terminaux, N un ensemble de non-terminaux disjoints de Σ, $S \in N$ est une variable spéciale de départ appelée *axiome*, et P est un ensemble de règles de production de la forme $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta \in (\Sigma \cup N)^*$. Dans le cas des langages visuels, les terminaux sont des primitives visuelles (des traits, par exemple) et les non-terminaux des objets abstraits (un triangle, par exemple).

Grammaires positionnelles L'évolution des travaux dans ce domaine peut être rapidement résumée en regardant les modifications apportées aux règles de production dans les extensions proposées. L'intégration de relations spatiales entre les objets composés a été la première évolution notable, et les travaux de Orefice *et al.* [88] proposent d'écrire les règles de production de la façon suivante :

$$A \rightarrow X \ \mathrm{REL}_1 \ Y \ \mathrm{REL}_2 \ Z$$

Les auteurs ont alors proposé un mécanisme permettant analyse de type *LR* [6] de ces grammaires. Ici, REL_1 et REL_2 peut signifier « vers la gauche », « vers le haut », etc. En compilation classique, la seule relation possible est « vers la droite » et est implicite.

Grammaires de graphes Une première généralisation de ces approches a consisté à représenter les objets sous forme de graphes, et non plus des primitives visuelles simples, comme des traits par exemple. Les travaux de Costagliola *et al.* [21, 22] proposent des règles de production de la forme suivante :

$$\alpha \rightarrow \beta E$$

où β correspond à un sous-graphe qui peut être transformé en un nouveau sous-graphe α. E représente la transformation des connexions entre les sous-graphes et le graphe hôte. La nécessité de générer un graphe avant le début de l'analyse empêche l'application de cette approche à des documents anciens, à cause de l'ambiguïté des relations possibles et du bruit présent.

4. Afin de ne pas alourdir démesurément ce manuscrit, nous ne préciserons pas davantage ici ces notions qui sont largement introduites et détaillées par Aho, Sethi et Ullman [3].

5. Nous introduisons ici quelques notations qui pourront aider le lecteur à mieux comprendre nos propositions dans la partie II.

6. On qualifie généralement d'*ascendants* les analyseurs syntaxiques procédant par regroupements progressifs de terminaux et non terminaux jusqu'à production d'une interprétation globale unique (l'axiome) ; et de *descendants* les analyseurs partant de l'axiome pour développer progressivement les règles de production et essayer de trouver des terminaux qui confirmeront la structure prédite. Dans le premier cas, les analyseurs syntaxiques sont de type $\mathrm{LR}(k)$ où k est le nombre de symboles qui peuvent être considérés avant de prendre une décision de réduction. Dans l'autre cas, on parle d'analyseurs de type $\mathrm{LL}(k)$. Ces notions sont définies pour les langages séquentiels hors-contexte.

Grammaires attribuées de multi-ensembles Une approche plus récente est celle proposée par Marriott *et al.* [76], munissant les objets d'attributs [55] pour véhiculer des propriétés graphiques et permettre la validation de contraintes géométriques. Les règles de dérivation prennent alors la forme suivante :

$$A \rightarrow \alpha \text{ where } C\,F$$

où α est un ensemble d'objets, C un ensemble de contraintes géométriques sur les caractéristiques de ces objets, et F un ensemble d'expressions permettant de produire les attributs de A à partir de ceux de α. Marriott *et al.* fournissent alors l'exemple suivant de règle pour décrire l'organisation spatiale d'une division mathématique :

$$\begin{aligned} \mathit{DivTerm} \quad \rightarrow \quad & \mathit{Exp}_1\ \mathit{Exp}_2\ \mathit{hbar} \text{ where} \\ & \mathit{Exp}_1.\mathit{xmin} > \mathit{hbar}.\mathit{xmin}\ \& \\ & \mathit{Exp}_1.\mathit{xmax} < \mathit{hbar}.\mathit{xmax}\ \& \\ & \mathit{Exp}_1.\mathit{ymin} > \mathit{hbar}.\mathit{ymax}\ \& \\ & \mathit{Exp}_2.\mathit{xmin} > \mathit{hbar}.\mathit{xmin}\ \& \\ & \mathit{Exp}_2.\mathit{xmax} < \mathit{hbar}.\mathit{xmax}\ \& \\ & \mathit{Exp}_2.\mathit{ymin} < \mathit{hbar}.\mathit{ymax} \\ & \mathit{DivTerm}.\mathit{xcenter} = \mathit{hbar}.\mathit{xcenter} \\ & \mathit{DivTerm}.\mathit{ycenter} = \mathit{hbar}.\mathit{ycenter} \\ & \mathit{DivTerm}.\mathit{meaning} = (\ \mathit{Exp}_1.\mathit{meaning} \,/\, \mathit{Exp}_2.\mathit{meaning}\ ) \end{aligned}$$

Ce type de formalisme permet l'expression de relations spatiales et sémantiques entres les éléments composant un objet abstrait, et la génération automatique d'un système recherchant des primitives visuelles vérifiant ces relations. Toutefois, il impose une analyse guidée par les données (« ascendante ») inadaptée au problème d'interprétation de documents anciens ou dégradés.

Grammaires logiques Pour terminer, notons que dans le domaine de l'interprétation de documents numérisés, la seule approche intégrant à la fois une dimension grammaticale et une dimension logique est l'approche DMOS proposée par Coüasnon [23], que nous présenterons dans la sous-section 2.3.3 car elle dispose d'autres avantages.

Des techniques connues existent cependant en compilation classique, et on peut notamment citer l'exemple des *Definite Clause Grammars* (DCG), proposées par Pereira et Warren [95], qui permettent de construire rapidement des compilateurs guidés par un but capables de faire face à des langages contextuels.

2.3.2.4 Un problème à la croisée de plusieurs domaines scientifiques

Nous constatons qu'au cours de ces vingt dernières années, beaucoup d'efforts ont été consacrés à la structuration des systèmes d'analyse afin de permettre une conception et une adaptation plus faciles pour des types de documents variés, ainsi que la production d'interprétations sémantiques organisées et pas simplement « à plat ». Les approches performantes proposent alors de séparer les connaissances à priori, exprimées de façon déclarative, de la stratégie de recherche d'une solution dans un espace complexe.

Le travail de formalisation et l'adaptation de travaux issus du domaine de la compilation ont permis de systématiser la production d'outils d'analyse disposant de propriétés connues. Toutefois, la gestion de l'imperfection de la connaissance et des données est généralement

insuffisante pour permettre l'analyse de documents anciens dégradés, et les reproches formulés par Mao *et al.* [72] à cet égard restent valables en ce qui concerne la dissociation des étapes de segmentation et d'interprétation, ainsi qu'au niveau de la gestion du bruit et des ambiguïtés (voir sous-section 2.1.3). Rares sont les approches permettant de réaliser une interprétation contextuelle des formes primitives, ce qui est pourtant nécessaire pour saisir le sens de nombreux documents, comme montré par Watanabe *et al.* [123], ainsi que Marriott et Meyer [75].

Au niveau des approches formelles (principalement grammaticales), trois reproches peuvent être faits au regard de notre problématique.

1. Tout d'abord, elles sont généralement basées sur des approches ascendantes qui, selon Coüasnon [24], apportent une confirmation syntaxique à une segmentation préalable, sans exploitation du contexte.
2. Par ailleurs, elles s'intéressent à la reconnaissance de formes très primitives (comme des composants de schémas électriques, par exemple) pour lesquelles les approches statistiques offrent déjà de bons résultats.
3. Finalement, lors de l'interprétation de documents anciens, il s'agit de détecter et d'interpréter les fragments utiles d'une image dégradée d'un langage visuel partiellement connu, et non de valider et transformer la totalité d'une phrase visuelle bien identifiée d'un langage clairement défini vers un autre domaine de représentation.

L'approche diamétralement opposée pourrait donc être de s'intéresser aux techniques d'analyse de scènes, qui travaillent sur des données très bruitées, et pour lesquelles les analyses descendantes (guidées par un but précis) semblent apporter de bons résultats [49] en complément de mécanismes ascendants.

À partir d'une influence prononcée des techniques de traitement de signal, on constate donc un glissement des problématiques d'analyse et d'interprétation d'images de documents vers des considérations proches de la compilation avec gestion des erreurs, de l'analyse de scènes, voire de l'analyse de langues naturelles avec les approches plus formelles.

2.3.3 Approches déclaratives extensibles

La grande variabilité des difficultés qui peuvent être rencontrées lors de l'interprétation de documents anciens ou dégradés semble rendre impossible la proposition d'une approche générique qui figerait les relations spatiales, les formes de contexte, ou des objets manipulés.

L'approche DMOS (pour « Description et MOdification de la Segmentation »), proposée par Coüasnon [26, 23, 24], apparaît alors comme un compromis efficace car elle propose un langage de description EPF *(Enhanced Position Formalism)*, basé sur une extension à deux dimensions des *Definite Clause Grammars* [95, 90] et un moteur d'analyse qui peuvent être utilisés et étendus pour être adaptés à l'interprétation de nouveaux types de documents.

Le système de base a en effet été spécialisé et utilisé pour reconnaître des documents récents et anciens, toujours dégradés, tels que des partition musicales, des registres matricules, des décrets de naturalisation, des tableaux complexes, et bien d'autres [29]. Certains des systèmes produits ont été utilisés pour analyser plusieurs centaines de milliers de pages [30].

C'est, à notre connaissance, la seule approche de ce type. Elle est fortement influencée par l'analyse de langues naturelles. Générique, elle permet après la compilation de la description de document de produire un système qui exploitera l'analyseur ainsi produit, les mécanismes généraux de recherche de solution, ainsi que des détecteurs de formes primi-

tives (traits, composantes connexes, ou autres classifieurs disponibles) pour interpréter une image donnée.

Cette approche propose donc une gestion relativement efficace du bruit, grâce à une analyse guidée par un but, et la possibilité de guider la stratégie de recherche au niveau des terminaux et des non terminaux. La gestion des ambiguïtés est gérée automatiquement par le mécanisme de retour arrière. La connaissance à priori est séparée du mécanisme d'analyse qui peut évoluer indépendamment. Les relations de position, les relations contextuelles, ainsi que la construction de la structure résultat sont exprimées de façon déclaratives à l'aide du langage logique sous-jacent. L'expressivité du langage et sa capacité à être étendu ont, de façon prévisible, donné lieu à plusieurs extensions notables :

- l'intégration de mécanismes perceptifs pour améliorer la détection de traits effacés [59] et de lignes de texte [60] ;
- un analyseur dédié aux tableaux complexes et dégradés [77] ;
- ou encore l'intégration de mécanismes statistiques pour évaluer la fiabilité des éléments détectés [73].

Cependant, cette approche nécessite une écriture des grammaires EPF par un expert, à la fois du système et du type de document à traiter. L'intégration de mécanismes d'adaptation automatique du système aux données semblerait donc nécessaire pour limiter ce travail fastidieux.

2.4 Conclusion

Au terme de ce deuxième chapitre, nous constatons que le travail d'interprétation de documents (anciens en particulier) se place à l'intersection de nombreux domaines scientifiques qui proposent une palette de méthodes. Il est donc beaucoup plus judicieux de réutiliser et étendre ces méthodes plutôt que de chercher à développer intégralement un nouveau système d'interprétation de fonds documentaires.

Ce chapitre nous a donc permis d'identifier une base de travail sur laquelle nous pourrons construire des propositions nouvelles, en identifiant les leviers d'amélioration possibles. Il est donc possible de lister les propriétés d'un outils d'interprétation de pages isolées (page par page, ou image par image) qui nous servira ensuite de référence. Nous proposons d'appeler cet outil « module d'interprétation de page », et la définition que nous en donnons ici est informelle ; nous formaliserons ces éléments dans les chapitres 5 et 6. Voici la liste des propriétés de ce module d'interprétation de page qui nous semblent essentielles pour construire un système d'interprétation de fonds documentaires.

1. Il doit permettre une séparation claire de la *connaissance à priori*, définie lors de la phase de *conception* par un *expert*, et du *mécanisme automatique d'appariement* entre le modèle de document ainsi exprimé et le contenu de l'image, utilisé lors de la phase d'*exécution*.

2. La connaissance à priori doit être exprimée de façon *déclarative* et *extensible*, de façon à permettre la génération automatique d'un système spécifique à un type de document grâce à des outils (langage de description, compilateur, algorithme de traitement de l'image, etc.) génériques.

3. Le mécanisme d'appariement entre le modèle de document et le contenu de l'image doit être *guidé par le modèle* (ou « par le but ») afin de permettre une gestion systématique du bruit, de l'ambiguïté, et plus généralement de la difficulté de la recherche

d'une solution dans un espace vaste grâce à des techniques de filtrage, de retour arrière, de prédiction, etc.

Nous supposerons que ce module d'interprétation de page dispose de fonctions permettant l'extraction de primitives visuelles, et nous ne nous préoccuperons pas davantage de traitements au niveau du signal de l'image.

Ces caractéristiques ressemblent évidemment à celles des systèmes produits avec la méthode DMOS. Notons dès à présent que si nos propositions ont permis l'extension effective de cette méthode, nous nous sommes efforcés d'identifier précisément les modifications réalisées pour permettre une réutilisation facile de nos travaux. Il sera alors possible de s'affranchir de certaines de ces propriétés que nous venons de lister ; certaines de nos propositions pouvant, par exemple, être exploitées dans des approches où la connaissance est exprimée de façon algorithmique.

Les prochains chapitres de cette première partie s'intéresseront à identifier les leviers d'amélioration d'un tel module d'interprétation de page. Notre objectif principal sera alors de comprendre comment améliorer la gestion des problèmes survenant lors de l'interprétation (pendant l'exécution) grâce à l'utilisation de connaissances supplémentaires à celles exprimées à priori par un expert, ou extraites de l'image. Nous chercherons à permettre au module d'interprétation de page de tirer profit d'informations contextuelles issues du fonds documentaire (dans le chapitre 3) ou fournies par un opérateur humain (dans le chapitre 4), ce qui nécessitera de comprendre comment l'intégrer dans un système plus général dans lequel il s'intégrera.

Finalement, il est intéressant de remarquer qu'en l'état actuel des choses, si les approches statistiques sont difficiles à utiliser dans le cas de documents anciens, c'est à cause de la nécessité de fournir à l'apprenant des exemples pour amorcer le système, exemples particulièrement coûteux à produire. La possibilité de diminuer le coût de correction nécessaire pour la production de résultats corrects est donc un enjeu important pour permettre d'utiliser efficacement ces approches et s'en servir comme levier pour augmenter la performance globale du système.

Chapitre 3

Interprétation contextuelle de fonds documentaires

Introduction

Ce chapitre s'intéresse à la possibilité de tirer profit d'informations qui ne sont pas directement accessibles au sein d'une image isolée, mais qui proviennent de l'ensemble d'images du ou des documents traités. Comme nous l'avons déjà mentionné dans le chapitre précédent, l'état de l'art tend à montrer qu'il est pertinent d'essayer d'interpréter les images de document en considérant leur environnement (voire en interagissant avec celui-ci, mais nous y reviendrons). Le contexte documentaire tel que nous l'avons défini au chapitre 1 constitue donc une partie de cet environnement.

Dans ce chapitre, nous cherchons donc à identifier *quelles sont les connaissances* liées au contexte documentaire, *comment les exprimer,* et *comment les exploiter.* Il sera intéressant de distinguer le travail réalisé lors de la *conception* de chaque système, et celui effectué automatiquement par ce dernier lors de son *exécution*.

Nous basons notre analyse sur le constat fait par Saund [106], selon lequel les systèmes de traitement de documents visant une utilisation en production doivent faire face à trois catégories de problèmes :

La classification des documents par type Ce problème *(Doctype Classification)* consiste à savoir à quel type de document appartient une image de page.

L'identification du rôle fonctionnel d'éléments visuels Ce problème *(Functional Role Labeling)* consiste à déterminer le statut d'éléments textuels et graphiques relativement au modèle de la structure d'un document.

L'analyse d'ensembles de documents Ce problème *(Document Sets)* consiste à exploiter les relations existant entre les pages et entre leurs contenus au sein d'un regroupement cohérent.

Dans le cas des documents anciens qui nous intéresse, nous avons vu dans le chapitre précédent qu'il était nécessaire de guider l'interprétation des images des pages à l'aide d'un modèle afin de permettre la localisation et l'interprétation des éléments pertinents. Ce travail d'interprétation correspond à ce que Saund appelle « identification du rôle fonctionnel » et nous allons, dans ce chapitre, nous intéresser à l'intégration de ce processus dans le travail plus large d'interprétation d'un fonds documentaire.

Nous verrons dans la première section de ce chapitre que cette étape d'*interprétation* est fortement liée à la capacité à *détecter le type de document*, et nous décrirons alors les architectures et caractéristiques des approches de la littérature qui mettent en œuvre ces processus. Ce premier temps sera l'occasion de présenter les méthodes qui permettent d'éviter de subir le volume de données à traiter, notamment grâce à une certaine adaptation des différents modèles utilisés.

Nous nous intéresserons ensuite dans la seconde section de ce chapitre aux méthodes permettant de faire du volume de données un atout. Les approches concernées exploitent les *redondances* et les *relations* qui peuvent exister entre les données des différentes pages, et offrent des réponses au problème d'analyse d'*ensembles de documents*.

3.1 Faire face à la variabilité

L'interprétation de fonds documentaires volumineux nécessite de pouvoir faire face à plusieurs difficultés. La première est liée à la quantité d'information : il n'est pas rare qu'un fonds d'archives contiennent plusieurs dizaines de milliers d'images, et occupe un espace disque de plusieurs centaines de gigaoctets. La seconde est liée à la variabilité des contenus présents dans les images à traiter, ainsi qu'aux différentes mises en formes des documents : il faut être capable, d'une part, de distinguer les types de documents à traiter, et d'autre part, de définir un modèle de document adapté à chacun de ces types.

3.1.1 Automatiser la gestion du volume

Une des approches courantes dans la communauté scientifique du traitement de documents est, de la même manière que dans de nombreuses communautés scientifiques, de s'appuyer sur des systèmes facilitant la construction d'enchaînement de traitements (ou *workflows*) pour construire des outils généralement destinés à évaluer la performance d'algorithmes sur une tâche précise.

La plate-forme DAE proposée par Lamiroy et Lopresti [56] est un exemple de ces approches qui permettent à la communauté de bénéficier d'outils et d'une place commune pour comparer différentes méthodes. Ce système permet la définition d'enchaînements de traitements qui vont consommer les données d'un jeu de test connu, grâce aux outils mis à disposition, et produire des résultats qui pourront être comparés à la vérité terrain du même jeu de test. Toutefois, ce système ne propose aucun mécanisme permettant l'exploitation du contexte documentaire qui permettrait de faire face aux problèmes liés à l'interprétation d'un fonds documentaire ancien ou dégradé.

À l'instar des systèmes d'enchaînement de traitements à but expérimentaux, dont Yu et Buyya [127] proposent une taxonomie intéressante, la plate-forme DAE ne semble pas orientée vers un objectif de production. On peut cependant noter l'intérêt de faciliter la définition d'enchaînements de traitements modulaires, consommant les données de sources bien définies. Les efforts de standardisation des interfaces des modules d'analyse, la capacité à sélectionner des sous-ensembles des données à traiter (lot d'entraînement ou de test, données brutes ou vérité terrain, etc.), ainsi que la tolérance aux fautes, forment la base de systèmes rigoureux pour l'analyse de documents.

Il n'est alors pas à exclure qu'un système d'interprétation de fonds documentaires puisse être implémenté à l'aide d'un système de construction de *workflow*, mais à notre connaissance rien de tel n'est proposé actuellement : il est donc nécessaire de pouvoir mettre en place des solutions plus spécifiques pour permettre le traitement de fonds documentaires

anciens ou dégradés. Ceci impose de savoir quelles connaissances exploiter, et comment exprimer ces dernières.

Plus généralement, depuis quelques années, de nouveaux modèles d'analyse de données en grand volume ont émergé, conséquences du développement important des moteurs de recherche pour le Web. Parmi ces nouvelles approches, on peut citer le modèle de calcul *map-reduce* [31] qui permet de traiter des volumes considérables de données en minimisant le volume d'informations échangées sur le réseau d'interconnexion des machines de calcul. Les implémentations classiques de ce modèle permettent, en outre, une très bonne tolérance aux fautes. De nouvelles organisations de bases de données [16] permettent également de manipuler avec plus de facilité des données structurées (avec des schémas facilement extensibles) et en grand volume. Ces nouveaux modèles semblent prometteurs pour le traitement d'images de documents, mais n'ont, à notre connaissance, pas encore été utilisés pour construire un système complet.

3.1.2 Sélectionner le traitement selon le document

Dans le cas du traitement de documents, la variabilité des données à analyser impose d'être capable de déterminer le type de chaque image pour choisir automatiquement la suite des traitements à lui appliquer.

3.1.2.1 Classifier puis reconnaître

Pour déterminer quelles opérations d'extraction et de validation il convient de réaliser sur une image ou un ensemble d'images, les systèmes de gestion automatique de documents administratifs donnent des exemples d'architectures et de comportements adaptés à des environnements de production.

Le système *smartFIX* [33, 52, 53] est particulièrement intéressant car, après un développement académique, il a bénéficié d'une industrialisation poussée qui a fait de cet outil une référence, en particulier grâce à sa capacité à analyser des formulaires de soins médicaux pour l'automatisation des procédures de remboursement. Son architecture permet de dissocier plusieurs processus essentiels au fonctionnement du système, associés à des expertises différentes. On peut distinguer :

Le gestionnaire de documents Ce composant permet la définition de scripts d'identification de documents, puis d'extraction et de validation des informations utiles : contraintes numériques, interrogation d'une base de donnée, etc. *Cet outil est utilisé par des experts pour concevoir des traitements spécifiques.*

L'analyseur Ce processus exploite les définitions de documents pour analyser chaque nouvelle image. Il commence par détecter le type de document concerné, puis lance le script d'extraction associé. Enfin, il effectue des contrôles de cohérence interne au document et des optimisations statistiques pour maximiser localement la vraisemblance de l'interprétation grâce à un ensemble de contraintes entre les contenus.

Le serveur d'association Ce composant permet de valider certaines règles spécifiques au problème considéré, comme par exemple, dans le cas des formulaires de soins, que le numéro d'assuré détecté correspond bien au nom reconnu.

Le vérifieur Ce processus permet de faire valider ou corriger par des opérateurs humains les éléments peu fiables ou incohérents.

Les bases de données Ces composants permettent de stocker les différentes informations utiles au cours du cycle d'analyse de chaque document. Elles fonctionnement globalement en cascade : la base contenant les éléments à traiter est alimentée en continu, et les processus de traitement consomment ces éléments pour alimenter la base d'éléments à contrôler. Le contrôle (automatique ou manuel) permet de déplacer les éléments validés vers la base d'export permettant la connexion avec d'autres outils, comme le système de paiement dans le cas des formulaires de soins.

Le contrôleur Finalement, ce composant est chargé de la coordination des processus du système, et permet le pilotage de ce dernier.

On peut noter qu'en tant que système commercial, *smartFIX* bénéficie d'une attention particulière à la qualité des données produites. Les mécanismes de rejet des éléments inconnus sont alors particulièrement importants pour éviter l'interprétation erronée de nouveaux types de documents, et les opérations de classification des types de documents doivent donc être définies avec soin. La validation de contraintes métier, le recours à des opérateurs humains et l'architecture du système en font une approche robuste, qui permet, par ailleurs, d'estimer automatiquement la qualité des données produites [110].

Cependant, comme le fait remarquer Belaïd [12], ces systèmes sont dédiés à l'analyse en flux avec l'arrivée continue de nouveaux documents. La nécessité de minimiser le temps de traitement de chaque document impose alors un fonctionnement très linéaire à ce type d'approche. L'objectif de ces systèmes est donc différent de celui visé lors de l'interprétation de documents anciens organisés en lots bien identifiés, et qui n'imposent pas de telles contraintes temporelles aux traitements. Bien que l'architecture des systèmes d'analyse de documents administratifs soit intéressante, la temporalité et la linéarité de leurs traitements limitent les croisements d'informations possibles entre pages.

Dans un contexte aux contraintes proches, la *numérisation massive* de livres à des fins d'indexation (pour des moteurs de recherche sur le Web, en particulier) obéit à ce même impératif de minimiser le temps de traitement d'un ouvrage. Coyle [28] oppose alors cette *« mass digitization »* aux initiatives visant à produire des contenus pour les bibliothèques numériques nécessitant une structuration et une fiabilité de l'information bien plus élevée que celle requise par les moteurs de recherche.

L'interprétation de documents anciens nécessite donc de considérer des fonds documentaires cohérents, et de définir une nouvelle temporalité dans les traitements afin de s'affranchir de la linéarité qui semble empêcher l'exploitation du contexte documentaire : il faut permettre la mise en place d'un échange entre le niveau de la page et un niveau global. Ceci étant, les systèmes industriels présentent des caractéristiques tout à fait intéressantes face aux contraintes liées au volume de données.

3.1.2.2 S'adapter à la structure documentaire

Dans une optique radicalement différente, le système *ArCAnOID* proposé par Lladós *et al.* [64] vise clairement la production de contenus pour des bibliothèques numériques, en vue de permettre leur utilisation et leur enrichissement collectif par des experts, à l'instar de la plate-forme proposée par Coüasnon *et al.* [27].

L'organisation du système *ArCAnOID* est entièrement tournée vers un traitement des documents en trois étapes visant à produire des informations extrêmement structurées et très abstraites. Le processus global de traitement des images est le suivant :

1. Les images sont d'abord traitées par le *module d'analyse*, qui contient une définition

du flot d'exécution (qui peut s'appuyer sur la structure du fonds documentaire pour déterminer les opérations à réaliser sur chaque image) et des outils d'analyse d'image. L'objectif est de localiser et de reconnaître des fragments d'images qui constitueront les éléments de base de la représentation globale du fonds documentaire. Dans un des exemples proposés par les auteurs, un cas d'archives de contrôles aux frontières, il s'agira de détecter les zones de texte et d'identifier les noms propres, ou les dates.

2. Ensuite, les éléments détectés sont systématiquement soumis au *système de correction et de validation* qui permet à des humains (non experts) de fiabiliser et compléter l'information produite. Il s'agit alors d'éliminer les pages inutiles, de corriger les résultats de l'OCR, etc. L'objectif est de produire des *termes* fiables qui seront utilisés dans la dernière étape.
3. La dernière étape, dite de *« capture de connaissance »* , consiste à permettre à des experts de former des *concepts abstraits* (famille, zone géographique, etc.) en reliant les *termes* grâce à des *relations* (« est fils de », etc.). Cette étape est complètement manuelle.

Le module de stockage des connaissances, composé de trois bases aux objectifs complémentaires, permet d'identifier les types d'informations utiles :

La base des contenus extraits des images Elle stocke les *termes* extraits des images.

La base d'organisation de la collection Elle stocke les informations relatives à la structure du fonds documentaire.

La base des informations sémantiques Elle stocke les concepts abstraits décrits par les experts.

Cette approche tire profit de l'organisation des documents en permettant l'analyse par lot de ces derniers, grâce à une connaissance préalable de l'organisation documentaire (ce qui n'exclut pas la nécessité de faire face aux imprévus). Elle autorise alors des regroupements de mots similaires, et autres techniques de croisement de l'information, selon les algorithmes utilisés par le concepteur du système de correction et de validation.

Finalement, même si cette approche ne semble pas utiliser la connaissance produite pour améliorer les traitements réalisés, principalement à cause de son fonctionnement très linéaire, on peut remarquer deux aspects originaux.

1. Le système propose d'intégrer toutes les étapes du cycle de vie d'un document sous forme numérique, depuis sa numérisation jusqu'à son exploitation par des experts. Ceci permet la mise en place d'un format de données unique, flexible, et évite les recopies entre différents systèmes.
2. Cette approche vise à permettre l'enrichissement progressif de la connaissance relative au fonds documentaire considéré. Ce mécanisme semble propice à l'exploitation des informations produites, ainsi qu'à l'ajout de nouvelles sources de connaissances.

Ici encore, il semble nécessaire de relâcher la contrainte de linéarité pour permettre une meilleure utilisation de l'information contenue dans le fonds documentaire.

3.1.3 Acquérir progressivement les modèles

Un des premières nécessités dans la gestion de données variées est de minimiser le coût de mise en œuvre d'un système versatile. Permettre à un système de construire progressivement des modèles de documents est un premier pas dans la mise en place de mécanismes

de rétroaction permettant d'augmenter l'impact de l'information extraite, puisqu'on peut alors imaginer retraiter les images et produire de meilleurs résultats, ou mieux traiter de nouvelles images à la structure similaire. L'idée d'apprentissage incrémental fait alors écho aux recommandations de Baird *et al.* [7] qui considèrent qu'une des fonctionnalités qui rendrait un système d'interprétation d'images de documents robuste serait de lui permettre de diminuer de façon monotone son erreur au cours du temps, en lui permettant de fonctionner et s'améliorer sans limite de durée.

Dans cette sous-section, nous nous intéressons aux techniques d'adaptation de modèles décrivant la structure des documents.

3.1.3.1 Mémoriser pour détecter des irrégularités

Une des méthodes les plus anciennes et les plus utilisées consiste à permettre au système d'interprétation de détecter un manque d'information, et de réclamer une correction soit des résultats (intermédiaires ou non), soit du modèle utilisé.

Le système de reconnaissance de formulaire intelligent proposé par Cheriet *et al.* [18, chapitre 2] présente différentes étapes d'une analyse qui permet de relier progressivement les contenus reconnus entre eux et de construire des objets de plus en plus abstraits. Chacune des étapes de cette analyse peut remettre en cause le travail des étapes précédentes, ainsi que le modèle de document si aucune interprétation cohérente ne peut être trouvée. Dans l'architecture de ce système, les auteurs proposent les phases suivantes pour l'analyse de formulaires (des chèques bancaires, par exemple) :

1. analyse de la mise en page et localisation des éléments à reconnaître ;
2. pré-traitement des zones de l'image à reconnaître et vérification de la présence d'éléments graphiques obligatoires ;
3. reconnaissance des caractères ;
4. post-traitement avec, en particulier, la validation croisée des éléments d'un formulaire.

Si les résultats sont incertains, ou que le type de document est inconnu, alors un expert est sollicité pour pallier le manque de connaissance.

Le principal intérêt de cette approche au regard du problème d'interprétation de documents anciens est la distinction de deux types de contenus mémorisés : d'un côté la *mémoire à long terme* conserve la connaissance acquise lors d'une phase d'apprentissage, et d'autre part la *mémoire à court terme* stocke les informations produites lors de l'analyse. Chaque module d'analyse peut alors consulter ces mémoires, et modifier la mémoire à court terme. C'est une forme d'interaction qui permet de réviser facilement certains résultats intermédiaires, à l'échelle de l'analyse d'une image. On peut donc s'interroger sur la possibilité d'utiliser un tel modèle à l'échelle d'un fonds documentaire. Il faut alors être capable de déterminer *quelles informations mémoriser*, et *quand réutiliser* ces dernières.

3.1.3.2 S'adapter aux nouvelles données : raisonnement à base de cas

Réviser la connaissance d'un système après la gestion d'un cas nouveau est un problème d'intelligence artificielle bien connu pour lequel le modèle de *raisonnement à base de cas (case-based reasoning)* donne un cadre théorique intéressant. Selon Belaïd *et al.* [12], ce type d'approche s'appuie sur une base de cas (un ensemble de couples $\{$ *problème/cas*, *solution*$\}$) modifiable, et respecte le schéma de fonctionnement général suivant :

1. construction d'une représentation du problème (vecteur, graphe, etc.) ;
2. recherche de cas similaires dans la base de cas ;
3. ajustement de la solution selon le cas utilisé (valeur d'attributs, etc.) ;
4. apprentissage : si nécessaire, la connaissance est augmentée ou révisée.

La travail de conception impose alors un choix de représentation du problème et la définition d'algorithmes appropriés pour les différentes étapes.

Ainsi, Belaïd *et al.* [48, 12] mettent en œuvre, grâce à une représentation à base de graphe de neurones et des méthodes de décision floues, un système capable de réaliser simultanément l'identification du type de formulaire représenté par une image, et l'étiquetage fonctionnel de zones d'intérêt.

Une approche plus ancienne, mais basée sur des idées similaires, est celle de Robadey *et al.* [104, 103, 47] qui permet également un étiquetage fonctionnel de zones, dans des documents de bonne qualité. La base de connaissance est, dans ce cas, formée de modèles qui associent une interprétation (solution) à une configuration spatiale (cas). Une configuration spatiale est ici un graphe d'objets visuels, caractérisés plus ou moins précisément, reliés entre eux par des relations spatiales. Lors de l'exécution, si le système propose une interprétation incorrecte pour un cas, la correction manuelle de l'interprétation permet de mettre à jour automatiquement la base de connaissance.

Pour ces deux approches, la nécessité d'extraire et segmenter une certaine nombre d'objets visuels avant le travail d'interprétation semble, cependant, rendre ces approches peu utiles dans le cas des documents anciens ou dégradés, pour lesquels la distinction entre segmentation et reconnaissance est délicate. Par ailleurs, les résultats produits par ces approches sont généralement assez peu structurés.

Malgré ces critiques, on peut persister à penser que ce type de fonctionnement est particulièrement intéressant pour limiter le coût de mise au point d'un système de traitement de documents. Une autre limitation empêche pourtant, en l'état, l'application de ces méthodes : il s'agit du coût de correction nécessaire à la production d'interprétations valides. Ma et Doermann [67] indiquent en effet que pour corriger une solution avant de permettre l'utilisation d'un processus d'apprentissage, dans le système qu'ils ont développé, il pouvait être nécessaire de : corriger des problèmes de sur- et sous-segmentation, redimensionner ou déplacer des éléments, supprimer les éléments erronés, ou encore modifier l'étiquetage de certaines zones. Par ailleurs, il est peu probable qu'un seul exemple de structure permette de construire un modèle fiable.

En conclusion de cette section, on peut alors indiquer que si les approches existantes proposent des architectures intéressantes, basées sur des mécanismes et des algorithmes éprouvés, la mise en place de mécanismes permettant l'amélioration progressive des modèles ne peut être faite en l'état. Il est en effet indispensable, au préalable, de trouver comment réduire l'effort nécessaire à la production d'interprétations valides qui puissent ensuite être utilisées par ces méthodes. Pour faire face à cette difficulté majeure, il est donc nécessaire d'avoir recours à des connaissances supplémentaires lors de l'interprétation pour diminuer ce coût de correction, sans augmenter le coût de conception.

3.2 Tirer profit du contexte documentaire

Au delà des contraintes imposées par le volume de documents à analyser, il peut être possible de tirer profit des propriétés d'un ensemble de documents. L'objectif est alors de

produire des interprétations de meilleure qualité en considérant l'ensemble des données, plutôt qu'en s'intéressant à chacune de façon isolée. Ceci correspond à l'idée de Nagy et Veeramachaneni [82] d'un *adaptive classifier* qui serait capable de considérer l'ensemble des données avant de prendre une décision locale.

Comme le souligne Saund [106], peu d'approches tirent réellement profit du contexte documentaire car la majorité des approches travaille à l'échelle d'une image. Cette section regroupe donc les quelques rares exceptions dont nous avons connaissance. Nous avons distingué les approches selon les propriétés qu'elles exploitent, et l'utilisation qu'elles en font.

3.2.1 Exploiter les redondances

Dans cette sous-section, les redondances auxquelles nous faisons référence sont les occurrences multiples de signes (lettres, mots manuscrits, etc.) au sein des différentes pages d'un fonds documentaire.

3.2.1.1 Optimiser la réponse globale

Un premier exemple de l'exploitation de techniques de regroupement d'éléments graphiques visuellement similaires est le travail de Kluzner *et al.* [54] visant à reconnaître le contenu textuel de livres allemands du XVIII^{e} siècle. Bien qu'imprimés, ces ouvrages présentent une variabilité assez importante dans la forme des symboles, et une analyse OCR simple produit de mauvais résultats. Ces auteurs ont donc proposé une approche qui prend en considération l'ensemble du livre pour permettre un regroupement des mots visuellement similaires, et exploite les clusters ainsi formés pour fiabiliser les hypothèses de reconnaissance associées à chacun des mots contenus. L'approche proposée permet un gain de plus de 20% dans la qualité des résultats.

Cet exemple montre l'importance de la capacité à disposer d'un niveau d'abstraction dans l'interprétation permettant une prise de décision globale, par exemple pour l'optimisation d'une réponse grâce à un contexte formé de plusieurs pages. Ceci nécessite, lors de la conception, d'identifier les éléments à regrouper et la méthode à utiliser.

3.2.1.2 Spécialiser le modèle localement

Il peut être intéressant de profiter de cette fiabilité supplémentaire pour améliorer les modèles utilisés. C'est ce que propose l'approche de Xiu et Baird [126] qui permet de spécialiser automatiquement les modèles de caractères et de lexique lors de la reconnaissance d'un livre entier.

Plus qu'une exploitation de la redondance, qui est un prérequis de ce type d'approche, ce système s'appuie sur un modèle linguistique (de fréquence des mots) afin de permettre une adaptation ciblée des éléments problématiques selon le schéma suivant.

1. Le système reconnaît l'ensemble du livre avec la combinaison des modèles graphiques et linguistiques initiaux.
2. Il identifie les zones de désaccord. Il s'agit : (i) soit d'un caractère dont la valeur a été contrainte par celle de ses voisins avec le modèle linguistique (ex : « `r h e` » transformé en « `t h e` » car « `h` » et « `e` » sont fiables et que le lexique contient le mot « *the* ») ; (ii) soit de mots pour lesquels le modèle linguistique augmente l'incertitude

globale d'un mot (ex : cas où « t h e » est transformé en « a b c » car le lexique contient « *abc* » mais pas « *the* »).

3. Le système propose des adaptations à l'un ou l'autre des modèles (modification d'un modèle de caractère, ou ajout d'un mot au lexique).
4. Il vérifie que la quantité de « désaccord » diminue dans l'ensemble du livre et accepte les modifications précédentes si c'est le cas.

Ces étapes d'adaptation peuvent être répétées. Le choix des éléments à modifier peut être fait automatiquement, mais il est également possible de solliciter un oracle externe (typiquement, un humain).

Ce type d'approche est une amélioration des approches en deux passes (ou plus) permettant l'auto-apprentissage, dès lors que des interprétations valident suffisamment de contraintes linguistiques. Le système Beatrix [57] est un exemple de ce mécanisme plus classique appliqué à la reconnaissance d'écriture manuscrite. Dans ce cas, trois processus interagissent :

1. un processus de reconnaissance qui produit des hypothèses (même peu fiables) ;
2. un processus de validation linguistique (lexicale, syntaxique ou autre) des hypothèses qui permet d'identifier des éléments réellement fiables ;
3. un processus de d'adaptation du processus de l'item 1 s'appuyant sur les éléments considérés comme fiables.

Cette approche plus courante s'appuie sur un relâchement progressif des contraintes linguistiques, et fait l'hypothèse que les interprétations considérées comme très fiables ont peu de chance d'être le fruit du hasard.

Les approches visant à spécialiser le ou les modèles aux données en cours d'analyse offrent donc d'intéressantes perspectives pour l'amélioration de systèmes existants dès que des interprétations fiables peuvent être proposées — ce qui est, rappelons-le, la difficulté principale dans le cas de documents anciens dégradés mais structurés. En outre, on peut observer que si l'interaction avec un humain peut être réduite, elle peut également être optimisée car ces approches permettent de sélectionner l'information la plus utile à apporter.

3.2.2 Relier les données

Dans cette sous-section, nous nous intéressons à un autre type de propriété utile dans un fonds documentaire. C'est le cas, par exemple, des numéros de page sur un livre : même s'ils sont partiellement ou complètement effacés, il suffit de consulter quelques pages pour rapidement retrouver les bonnes valeurs. Les approches que nous présentons ici exploitent les relations sémantiques entre les contenus à reconnaître.

3.2.2.1 Optimiser l'interprétation

Un exemple d'utilisation des liens inter-pages permettant d'améliorer les hypothèses de reconnaissance individuelles des éléments est la reconnaissance de livres proposée par Lin *et al.* [63]. Ces auteurs proposent de croiser les hypothèses de reconnaissances des éléments détectés dans la tables des matières avec celles des titres de section détectés dans le reste du livre. Bien qu'elle suppose une localisation préalable fiable des contenus, cette approche illustre une idée simple qui permet d'augmenter la fiabilité des éléments reconnus, tout en construisant une structure globale permettant une navigation, basée sur des liens sémantiques, dans l'ensemble du document.

Saund [106] indique que ce type d'optimisation repose sur le schéma suivant emprunté à la reconnaissance de la parole et de l'écriture manuscrite :

1. sursegmenter les éléments à reconnaître pour former des unités atomiques (ex : des vignettes de numéros de pages) ;
2. regrouper les unités pour former un espace d'hypothèses pour la formation d'unités plus grandes (ex : former une séquence de numéros devant se succéder selon une contrainte d'incrément de 1) ;
3. estimer la vraisemblance des hypothèses individuelles selon un critère choisi (ex : score de reconnaissance des caractères avec un système donné) ;
4. effectuer une recherche dans l'espace des hypothèse pour trouver l'interprétation globale de vraisemblance maximale (ex : utiliser l'algorithme de Viterbi pour trouver la séquence la plus probable).

Ce type d'approche nécessite, lors de la conception, de déterminer les éléments à extraire, les relations qui les relient, et la méthode permettant de proposer une interprétation globale.

3.2.2.2 Découvrir automatiquement les règles de présentation

Une autre approche intéressante consiste à inverser le problème de reconnaissance pour se baser uniquement sur les contraintes reliant les éléments entre eux (comme l'incrément des numéros de pages ou de la numérotation des titres de section) pour découvir automatiquement les éléments respectant ces propriétés fonctionnelles, sans avoir à connaitre leur propriétés graphiques à priori. C'est ce que proposent Déjean et Meunier [32] afin de s'affranchir du travail de spécification des caractéristiques graphiques des différents titres d'un ouvrage.

Cette méthode se base sur une localisation préalable des éléments pouvant potentiellement appartenir aux différentes séquences recherchées. Ensuite, grâce à une reconnaissance (même peu fiable) du texte des éléments localisés, le système construit des séquences d'éléments à la mise en forme et aux contenus similaires (même schéma de numérotation). Lorsque des séquences suffisamment longues et fiables sont détectées, elles peuvent être étiquetées selon le type de contenu considéré, et les propriétés graphiques des éléments de ce type peuvent être déduites des exemples ainsi produits. Une seconde phase d'extraction d'éléments vérifiant ces propriétés graphiques permet alors de récupérer des éléments manqués lors de la première passe. Les auteurs ont également appliqué cette méthode pour la détection et l'optimisation des contenus de tables des matières, à l'instar de l'approche de Lin *et al.* mentionné précédemment.

Pour conclure cette section, on peut noter que si les approches tirant profit des relations inter-pages d'un fonds documentaire sont rares, et présupposent souvent une segmentation idéale, elles proposent des schémas qui peuvent être adaptés au cas des documents anciens, à condition de permettre la mise en place d'un mécanisme d'interprétation *global* et *non linéaire* au niveau du fonds documentaire, et d'autoriser une *réanalyse* des images afin de pouvoir remettre en cause les résultats du système à la suite d'éventuelles modifications des données locales.

Dans le cas des registres de ventes de la Révolution française, par exemple, s'il est possible, via un mécanisme d'optimisation quelconque, de changer la valeur du nom d'un ancien propriétaire pour « idem », alors il faudra être capable de produire un résultat final indiquant, pour la vente considérée, le véritable nom du propriétaire, et non « idem ».

3.3 Conclusion

Dans ce chapitre, nous avons passé en revue différentes méthodes et systèmes permettant, d'une part, de faire face aux contraintes de volume et de variabilité des fonds documentaires, et d'autre part, de tirer profit de l'organisation cohérente de ces données. Ces approches, relativement innovantes, donnent des outils pour diminuer le coût de mise au point du système, ou pour augmenter la qualité des résultats produits. Ceci est rendu possible par l'exploitation d'informations contextuelles extraites en considérant le fonds documentaire dans son ensemble plutôt qu'en interprétant chaque page de façon isolée.

Un système d'interprétation contextuelle de fonds documentaires doit donc être capable d'intégrer les points clés de ces approches. Nous proposons de les rappeler ici, afin d'établir une première version du cahier des charges des outils nécessaires à l'extension du module d'interprétation de page isolée, identifié en conclusion du chapitre précédent (page 37), qui sert de référence pour nos propositions.

Au niveau de l'*architecture du système*, tout d'abord, les approches de ce chapitre montrent qu'il est nécessaire de disposer, en plus du module d'interprétation de page centré au niveau d'une image, d'un *niveau d'abstraction global*, capable de :

1. *centraliser* les connaissances « factuelles » à propos du fonds documentaire : lexiques, organisation documentaire, données produites, etc au sein d'une base flexible ;
2. *gérer automatiquement* la circulation d'informations nécessaire à l'exploitation du contexte documentaire (connaissances « pratiques »).

Cette exploitation du contexte n'est possible que si le système autorise des modes de fonctionnement non-linéaires, afin de permettre la *remise en cause des résultats* produits, et l'*échange à double-sens* d'informations entre le niveau de la page et le niveau global. Ceci impose de pouvoir réinterpréter chaque page plusieurs fois au regard des informations extraites dans les autres pages. Nous proposons de qualifier ce mécanisme d'interprétation d'*« itératif »*.

En réintégrant la connaissance issue du contexte documentaire au sein du module d'interprétation de page, ce type d'approche doit permettre l'exploitation de liens sémantiques pour *fiabiliser les interprétations*, et *détecter automatiquement certains problèmes* : incohérence, manque, etc. Ceci peut alors permettre d'espérer une *diminution du coût de correction* des résultats, et un *sollicitation ciblée* des opérateurs humains. Il sera alors possible d'envisager l'utilisation de méthodes tirant profit de mécanismes d'apprentissage pour améliorer les modèles du système.

Concernant le *concepteur du système*, les outils génériques que nous souhaitons proposer doivent lui permettre :

1. d'*identifier* facilement les éléments qui doivent être considérés au niveau global ;
2. d'*exprimer* les relations et les contraintes sémantiques entre ces éléments, ou, à défaut, comment les *exploiter*.

Ceci doit se faire en minimisant la fragmentation de la connaissance et les perturbations du modèle de page qui doit rester unitaire.

Au terme de ce chapitre centré sur l'aspect contextuel de l'interprétation, nous constatons que la présence d'un opérateur humain au cours de ce processus est plus que jamais pertinente, car les liens sémantiques entre les éléments extraits de pages différentes permettent la détection de problèmes, et les redondances permettent d'accélérer le travail de correction. Nous verrons dans le chapitre suivant comment un système permettant l'interprétation de fonds documentaire peut bénéficier d'une communication avec un opérateur

humain au cours des traitements. On pourra remarquer qu'il s'agit finalement de mettre en place une double interaction : entre la page et son contexte, et entre le système d'interprétation et l'opérateur humain, chacune de ces formes imposant une certaine temporalité.

Chapitre 4

Interprétation assistée de fonds documentaires

Introduction

Humains et machines ont des capacités d'analyse et d'interprétation très différentes, et certains auteurs ont comparé leurs compétences dans le cadre de la reconnaissance d'écriture manuscrite [108] ou plus largement dans le cadre de la reconnaissance de documents [82]. Ces analyses mettent en avant les spécificités, et l'avantage évident, de l'intellect humain pour déchiffrer les messages produits par et pour l'homme : la gestion du bruit, la prise en compte d'un contexte extrêmement complexe, et des capacités d'adaptation et de déduction lui permettent de lire et comprendre des documents très dégradés. Les machines, quant à elles, disposent d'une capacité à résoudre des systèmes de contraintes de grande dimension, à ne rien oublier, et à réaliser très rapidement des tâches répétitives pour un coût de calcul devenu négligeable par rapport à celui du travail humain, qu'il convient donc de minimiser.

Nagy et Veeramachaneni [82] se sont intéressés aux caractéristiques d'une interaction efficace entre un système de classification et un opérateur humain. En particulier, ils suggèrent que *« l'opérateur ne devrait même pas avoir à regarder les données qui ont été classées sans problème »*, et également que *« chaque interaction devrait être utilisée par le système pour améliorer les tâches de classification subséquentes »*. Par ailleurs, ces auteurs déconseillent de confier la configuration fine d'un système à un humain pour préférer l'utilisation de mécanismes d'apprentissage ou d'optimisation généralement beaucoup plus efficaces.

Alors que la participation humaine se limitait traditionnellement à un développement du système et à sa configuration en amont des traitements, puis à la correction des erreurs produites en aval, certaines approches s'intéressent à son implication tout au long du fonctionnement du système, afin de tenter de diminuer la charge globale de travail.

La question se pose alors de savoir qui est assistant, et qui est assisté. Le titre de ce chapitre ne le précise volontairement pas, car selon les scénarios et les besoins, on peut constater deux tendances. Nagy et Veeramachaneni distinguent en effet les approches où les traitements sont *initiés par l'humain* de celles où les traitements sont *initiés par la machine*. Bien que ces auteurs préconisent les formes de traitements où la machine effectue le plus de travail possible et ne sollicite que très ponctuellement l'humain, ce genre d'approche n'est pas toujours possible dans le cas des documents anciens ou dégradés produisant beaucoup

d'erreurs, et nous consacrerons donc une section à chacune de ces deux formes.

Nous nous intéresserons aux rôles accordés à l'humain et aux tâches qu'il doit réaliser, aux phases du traitement au cours desquelles il peut intervenir, au comportement du système et aux conséquences de la présence de l'humain sur l'architecture de ce dernier. Nous nous intéresserons également, dans le prolongement des réflexions de Nagy et Veeramachaneni, à l'utilisation des informations fournies par l'humain : est-elle *durable*, c'est à dire qu'elle permet un guidage ou une remise en cause large des résultats du système ? Ou est-elle, au contraire, *éphémère*, c'est-à-dire ne sert-elle qu'à compléter ou corriger un élément ponctuel ?

4.1 Traitements initiés par l'humain

Nous nous intéressons tout d'abord aux systèmes fonctionnant selon un schéma où la machine assiste l'humain, et où ce dernier déclenche les opérations à réaliser, volontairement ou non. L'implication de l'humain peut alors lui faire prendre différents rôles, à plusieurs instants : au tout *début* du travail d'interprétation, vers la *fin*, ou *pendant* celui-ci.

4.1.1 Transfert de connaissances vers le système automatique

Le système AGORA proposé par Ramel *et al.* [98, 97, 100] est selon ses concepteurs [99] un outil visant *« à offrir à l'utilisateur (non expert) la possibilité de construire des chaînes de traitement permettant de faire évoluer progressivement le contenu de la représentation en fonction des caractéristiques des images à analyser et des objectifs visés »*. Le mécanisme d'analyse proposé par cet outil débute par une phase de pré-traitement visant (i) à distinguer les symboles du fond de la page ; et (ii) à produire un ensemble de composantes connexes à laquelle une étiquette (« texte », « graphique », etc.) est associée. Grâce à ces informations, l'utilisateur du système peut définir des règles qui permettront de transformer et classer les éléments graphiques selon leurs propriétés graphiques (forme, nature, position) et celles des éléments voisins.

Ce système est un premier exemple de système intégrant un opérateur humain dans son fonctionnement. Toutefois, cette approche limite l'interaction avec un humain à un transfert direct de connaissances sous une forme algorithmique qui seront ensuite appliquées directement sur un lot de documents. Ceci limite donc la phase d'interaction à une période de mise au point du système, et il faudra ensuite corriger les erreurs d'interprétation en post-traitement.

4.1.2 Correction libre en post-traitement

4.1.2.1 Correction distribuée

Un exemple d'approche faisant le choix délibéré de corrections libres en post-traitement est le système *ArCAnOID* proposé par Lladós *et al.* [64] déjà présenté en sous-section 3.1.2.2. Ce système vise à permettre un enrichissement très libre des informations relatives au contenu des images traitées grâce à un traitement en trois phases. La phase d'interprétation préliminaire extrait d'abord les éléments pertinents des images. Ensuite, une phase de correction et de validation permet à des opérateurs non experts de modifier très librement les contenus détectés et reconnus. Ils disposent d'un certain nombre d'outils d'édition, et peuvent appeler à la demande certains traitements automatiques (comme un module OCR),

afin de produire un ensemble de *termes* (entités visuelles localisées et annotées) avec une fiabilité élevée. Ceci permet à des experts, dans la dernières phase, de mettre en relations ces entités et former des concepts abstraits avec les éléments reconnus, comme, par exemple, pour indiquer que les personnes dont les noms apparaissent dans un même document sont de la même famille.

Ce type d'approche vise un niveau de qualité très élevé, et justifie donc le choix de corrections en post-traitement assez lourdes, entièrement à l'initiative des opérateurs humains. Par ailleurs, dans le cas d'*ArCAnOID*, les auteurs indiquent que les tâches de validation et de correction ne nécessitant pas d'expertise particulière, ces dernières pourraient être transmises à une communauté de bénévoles, à la façon des projets de saisie et correction collaborative de livres tels que le projet Guntenberg [1]. Cependant, on peut se demander s'il ne serait pas possible de tirer davantage profit des corrections faites par les opérateurs humains pour minimiser le coût manuel global.

4.1.2.2 Apprentissage de règles de correction

Toujours dans une optique de correction libre en post-traitement, le système WISDOM++ proposé par Malerba *et al.* [70, 71, 37] propose une approche originale consistant à apprendre comment corriger les erreurs de segmentation et d'interprétation d'un système. Ceci permet de rajouter un dispositif de correction automatique en sortie du système. Basé sur des mécanismes d'inférences de règles, WISDOM++ propose systématiquement à un expert de valider ou corriger les résultats fournis par un système donné, et grâce à un enregistrement des modifications effectuées, le système en déduit un ensemble de règles qui pourront être appliquées à d'autres problèmes similaires.

Le problème majeur de cette approche, au regard de notre problématique d'interprétation de documents anciens, est le découplage entre le système d'interprétation et le système de correction. Ceci limite fortement la complexité de l'interprétation qu'il est possible de mettre en œuvre, puisque les opérations de correction se limitent à des modifications de la segmentation, ou éventuellement des étiquettes associées à une structure à plat. Par ailleurs, cette méthode semble difficilement applicable dans un contexte comme celui des documents anciens ou dégradés, où le nombre d'erreurs est important, et les corrections manuelles fastidieuses. Il semble donc plus approprié de favoriser une exploitation permettant une remise en cause des résultats produits, à défaut d'essayer d'apprendre des structures complexes, afin de pouvoir diminuer la charge en correction manuelle.

4.1.3 Correction de résultats intermédiaires et guidage

4.1.3.1 Mise en place d'une communication asynchrone

Afin de permettre de tirer profit de la présence d'un humain au cours du travail d'interprétation, Bapst *et al.* [9, 10, 11, 8] ont proposé une architecture multi-agents permettant une analyse coopérative d'images de page. L'objectif de cette approche était de permettre la mise en place d'un environnement d'analyse concurrent, où les processus automatiques construisent (selon leur spécialité, à l'instar d'un *blackboard*) une interprétation structurée de l'image courante : structuration en blocs, paragraphes et mots ; reconnaissance du texte et de la fonte. L'utilisateur, de son côté, peut visualiser les résultats en cours de construction,

1. Le projet Gutenberg propose à la lecture plus de 40 000 livres libres de droits sous format électronique, saisis et validés par des bénévoles. Voir `http://www.gutenberg.org`.

et choisir de modifier des éléments de la structure ou de produire lui-même des éléments nouveaux, par exemple en localisant un bloc de texte manqué pour forcer sa segmentation en lignes puis sa reconnaissance.

Cette approche se base sur le constat de la nécessité de permettre des modes de fonctionnement variés, où l'utilisateur et le système automatique peuvent être aussi bien assistés qu'assistants. Ces auteurs indiquent également la nécessité, à cet effet, de définir clairement un *scénario* et un objectif d'analyse, et de permettre à l'humain d'interagir de façon *asynchrone* avec le système automatique afin de ne forcer aucune des deux parties à attendre. En outre, ils montrent également que si l'humain et le système automatique peuvent être interchangeables pour certaines tâches, alors ils faut garantir une *homogénéité* dans les données qu'ils produisent.

Malgré l'intérêt des constats faits par ces auteurs, le choix d'une architecture multi-agents semble avoir limité le développement de cette solution. D'une part, cette approche est basée sur une connaissance algorithmique rendant difficile le développement d'une solution cohérente produisant une structure résultat complexe. D'autre part, on peut supposer que l'environnement concurrent tend à créer des conflits entre l'humain et le système automatique qui doivent être anticipés dans le scénario global (Si l'humain supprime un élément, le processus automatique va-t-il le recréer ?) ou imposent de laisser le contrôle du système à l'utilisateur qui devient alors pilote de l'analyse, en décidant quels processus peuvent être activés à un instant donné. Finalement, la gestion manuelle des échanges d'information entre le système et l'utilisateur semble avoir été particulièrement lourde à mettre en œuvre.

4.1.3.2 Fusions d'indices et d'observations

Afin de faire face au problème de progression d'un système interactif, et au manque de formalisation de ce type de système dans la communauté de la reconnaissance de formes, Vidal *et al.* [120] ont mis au point un cadre théorique (bayésien) rigoureux qui permet la *fusion d'informations* entre les données extraites (observées) d'une forme à reconnaître (ligne de texte manuscrit [116, 117], mais aussi signal audio, texte à traduire, etc.) et celles fournies par un expert humain.

Ces auteurs se basent sur l'expression classique de l'hypothèse optimale d'interprétation décrite par l'équation suivante :

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h \,|\, x) \tag{4.1}$$

Ceci traduit le mécanisme suivant : étant donné une observation x (une séquence de graphèmes par exemple), le système dispose d'un modèle (graphique et lexical par exemple) permettant de formuler un ensemble d'hypothèses d'interprétation $\mathcal{H}$. Grâce à une évaluation de la probabilité de chaque hypothèse $h \in \mathcal{H}$, le système peut déterminer la meilleure réponse $\hat{h}$.

Pour intégrer la connaissance d'un opérateur humain, les auteurs proposent d'étendre ce modèle en permettant à un opérateur de fournir des informations f (pour *feedback*) après avoir consulté la donnée à reconnaître x et les hypothèses de reconnaissance h du système. L'équation 4.1 est alors enrichie de la façon suivante [2] :

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h \,|\, x, f) \tag{4.2}$$

2. Nous ne reprenons ici que les éléments fondamentaux de la proposition des auteurs. D'autres extensions sont décrites dans les publications citées.

Le système fonctionne alors de façon *incrémentale* : il utilise ces nouvelles informations pour proposer un nouveau résultat, et l'opérateur peut choisir de contraindre de plus en plus l'interprétation jusqu'à ce qu'elle soit satisfaisante. La figure 4.1 donne un exemple de dialogue possible entre un opérateur humain et un système chargé de reconnaître une ligne de texte manuscrit : l'opérateur ajoute progressivement des contraintes en validant le préfixe correct et en corrigeant les mots incorrects.

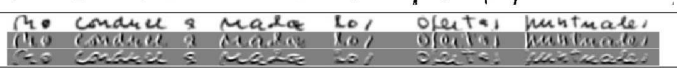
	Line Img							
	x							
INTER-0	(p)	()						
INTER-1	$(\hat{s})$	*(no*	*conduce*	*o*	*mala*	*los*	*afectos*	*eventuales)*
	$(\hat{s}_p)$	(no	conduce)					
	(c)			**(a)**				
	(p)	(no	conduce	a)				
INTER-2	$(\hat{s})$				*(nada*	*los*	*afectos*	*eventuales)*
	$(\hat{s}_p)$				(nada)			
	(c)					**(las)**		
	(p)	(no	conduce	a	nada	las)		
FINAL	$(\hat{s})$						*(ofertas*	*puntuales)*
	(c)							(#)
	$(p \equiv t)$	(no	conduce	**a**	nada	**las**	ofertas	puntuales)

FIGURE 4.1 – Exemple de scénario d'interaction pour la transcription de texte manuscrit. (Extrait de [116, figure 1].)

Ce type d'approche permet une *utilisation optimale* de l'information fournie par un opérateur humain, car elle utilise cette dernière pour calculer la nouvelle meilleure réponse selon son modèle (défini par le concepteur). Ceci donne une certaine garantie sur la *progression* globale de l'interprétation. Cependant, de façon assez paradoxale, l'information apportée n'est *pas forcément utile* pour le système : il n'y a aucune sélection de l'information à apporter pour minimiser le nombre d'itérations, et l'opérateur humain doit détecter lui-même les erreurs.

Au terme de cette première section, on peut indiquer immédiatement que la mise en place d'une interaction *pendant* la phase de traitement des données est nécessaire pour diminuer le coût de production des résultats, et pouvoir espérer, à terme, intégrer des mécanismes d'apprentissage dans un système destiné à l'interprétation de documents anciens et dégradés.

En effet, interagir avec un opérateur humain pendant cette phase permet au système de remettre en cause les résultats précédents et proposer de nouveaux résultats plus probables. Ceci nécessite un modèle de document capable de prendre en compte ces données externes et de les fusionner avec les données extraites de l'image.

Toutefois, maximiser l'impact de l'information fournie par l'opérateur humain n'est pas intéressant si cette information n'a aucune utilité pour le système. Il faut donc munir le système, lors de la conception, de mécanismes de détection d'erreur pour lui permettre de réclamer l'information dont un a le plus besoin. Ces mécanisme s'appuient généralement sur l'expression de contraintes entre les contenus d'une ou plusieurs pages, comme nous l'avons vu dans les chapitres précédents.

4.2 Traitements initiés par la machine

Dans cette section, nous considérons la seconde famille d'approches, à savoir celles qui proposent de laisser au système d'interprétation automatique la responsabilité de réclamer l'information dont il a besoin pour progresser auprès de l'opérateur humain.

En préambule, on peut faire deux remarques à propos de ce type d'approche.

1. On peut généralement considérer que le rôle donné à l'humain est celui d'*oracle*. Un *oracle* est un concept d'intelligence artificielle défini strictement [20], mais dans notre cas nous pouvons nous contenter de considérer qu'il s'agit d'un système externe fiable, capable de produire à la demande des résultats corrects.
2. Ces approches *initiées par la machine* doivent disposer de mécanismes de détection de problème ou de besoin d'information afin de solliciter l'étiquetage externe d'une information par l'humain.

4.2.1 Correction en post-traitement des erreurs détectées

4.2.1.1 Détection d'erreurs

Dans le cadre de la conception d'un système de traitement de formulaires, Cheriet *et al.* [18, chapitre 2], proposent un mécanisme centré sur une base de connaissance qui permet de déterminer si le formulaire à traiter est d'un type connu, et si les résultats produits sont fiables. Si le système constate un problème, il sollicite un expert humain capable d'ajouter un nouveau modèle de document à la base de connaissance, ou de corriger les résultats produits. La notion de *base de connaissance* est fondamentale, car elle permet une interaction *asynchrone*, ainsi que le stockage des informations fournies par l'humain pour une éventuelle exploitation ultérieure : exploitation du contexte, apprentissage, etc.

La capacité à *détecter des erreurs* est également essentielle pour permettre de solliciter l'humain pour un problème précis. Pour ce système de traitement de formulaires, elle se base sur les indices de confiances retournés par les modules de classification, et sur la vérification de présence d'éléments indiqués comme obligatoires lors de la conception (des bordures de cases, par exemple).

Dans cette approche, l'humain peut jouer le rôle d'oracle, voire d'expert capable de détecter les erreurs, et d'entraîneur selon qu'il corrige des résultats ou qu'il transfère sa connaissance de l'organisation des documents au système. Pour le reste, cette architecture ne permet pas à l'humain d'avoir une influence sur les résultats intermédiaires.

4.2.1.2 Contrôle des informations fournies par l'humain

Le système *smartFIX*, que nous avons déjà présenté en sous-section 3.1.2.1, permet un contrôle par des opérateurs humains de l'interprétation des documents administratifs traités. Lors de la phase d'analyse, les résultats peu fiables ou présentant des incohérences sont marqués pour révision manuelle. Ces problèmes sont détectés lors de la validation de propriétés sémantiques au sein du document (sommes de contrôle, présence d'éléments obligatoires, etc.) et en comparant les données extraites avec une base de données métier permettant un niveau d'interprétation supplémentaire. Dans le cas de formulaire de soins médicaux, par exemple, le système vérifie que le nom d'un assuré est connu et que le numéro reconnu lui correspond bien.

Dans ce contexte, le rôle des bases de données est primordial, car il permet la validation de contraintes complexes sur plusieurs pages et la mise en place d'une communication

asynchrone avec des opérateurs humains, en répartissant la charge de travail entre ces derniers. On peut alors parler d'interaction entre le système de reconnaissance et différentes sources externes d'information.

Afin de garantir la cohérence de l'interprétation finale, et limiter les erreurs des opérateurs humains, le système transmet automatiquement aux interfaces homme–machine un certain nombre d'informations relatives aux contraintes et liens entre les éléments du document, afin de permettre un *contrôle des informations saisies*.

La figure 4.2 montre un exemple d'interface de vérification du système *smartFIX* qui permet la validation et la correction d'un élément suspect. Grâce à plusieurs vues synchronisées, l'opérateur bénéficie d'une remise en contexte efficace où l'information est colorée selon son niveau de fiabilité, et où les structures détectées sont pré-remplies. L'export de contraintes du modèle de document vers l'interface est particulièrement intéressant car cette technique évite de dupliquer des connaissances entre le système d'analyse et les interfaces de validation. Des travaux récents menés par Forcher *et al.* [39] visent même à proposer automatiquement une explication compréhensible des raisons pour lesquelles le système sollicite l'opérateur, afin de permettre de mieux appréhender le problème détecté.

FIGURE 4.2 – Exemple d'interface de vérification du système *smartFIX*. (Extrait de [53, figure 9].)

Pour terminer, notons que la remise en cause des résultats permise par ce système ne semble possible que pour les éléments *terminaux* des structures résultats, comme les valeurs du contenu des cellules d'un tableau, sans permettre la remise en cause de la structure du résultat, ou de résultats intermédiaires : par exemple la structure du tableau, si une colonne a été oubliée. Ceci limite la remise en cause possible de l'interprétation réalisée, car les informations fournies n'ont qu'un impact très restreint sur la structure résultat.

4.2.2 Adaptation grâce aux exemples

Les systèmes s'appuyant sur un modèle de raisonnement à base de cas, que nous avons évoqués en sous-section 3.1.3.2, s'appuient également sur la disponibilité d'un oracle au cours de la phase d'interprétation. Ce dernier permet, lorsque le système lui soumet un nouveau résultat (final), de le corriger si nécessaire afin de fournir un nouvel exemple valide qui puisse être utilisé par le processus d'apprentissage.

Bien que certains systèmes, comme *2(CREM)* par exemple (voir page 44), permettent de détecter, dans une certaine mesure, les situations problématiques, l'humain doit tout de même vérifier la cohérence de l'interprétation produite, et corriger ou décrire l'intégralité d'une solution valide si c'est nécessaire. Dans le cas d'interprétations fortement structurées, c'est une tâche particulièrement fastidieuse, entièrement manuelle. On pourrait presque parler de fonctionnement initié par l'humain.

Ici encore, on s'aperçoit que les rôles d'entraîneur et d'oracle sont lourds à mettre en œuvre et nécessiteraient de permettre une remise en cause des résultats intermédiaires du système afin d'alléger le travail de correction avant de pouvoir envisager une quelconque forme d'apprentissage.

4.2.3 Correction de résultats intermédiaires

Afin d'éviter la propagation d'erreurs au cours de l'interprétation de données complexes, Ogier *et al.* [87, 85] ont conçu un système de reconnaissance de plans cadastraux qui permet de détecter automatiquement, en cours d'analyse, des situations problématiques. Le système est basé sur une stratégie de segmentations et de regroupements progressifs d'entités visuelles (traits, régions, etc.) en entités plus abstraites comme des parcelles, des quartiers ou des routes. La connaissance du système est exprimée à l'aide de règles, définies par le concepteur, qui régissent à la fois la composition des entités, mais également le contrôle de leur cohérence sémantique en exprimant les relations qui lient les entités ainsi formées (ex : une parcelle doit avoir un numéro dans la rue). Le système dispose également d'un ensemble de règle décrivant les « remèdes » aux problèmes qui peuvent être rencontrés. Ces règles peuvent demander une remise en cause de la segmentation et des traitements primitifs, si nécessaire.

Le système fonctionne alors selon des cycles tels qu'illustrés par la figure 4.3 : le système commence par extraire des primitives visuelles, puis il forme des objets grâce aux règles de composition. Il valide ensuite la cohérence du résultat partiel grâce aux règles sémantiques. Si un problème est détecté, il recherche une règle permettant de résoudre ce problème. Si aucune règle ne convient, l'expert humain est sollicité pour fournir une nouvelle règle de correction au système. Ensuite, le cycle d'analyse reprend.

Par rapport au système WISDOM++ qui cherche à apprendre des règles de correction d'erreurs de segmentation (présenté en sous-section 4.1.2.2), cette approche présente deux différences notables.

1. Elle sépare les connaissances relatives au document (composition, liens sémantiques) des règles de correction, ce qui permet la détection automatique de problèmes et la sollicitation de l'humain ;

2. Elle ne demande pas à l'humain de produire une solution (finale) correcte car elle ne permet pas l'inférence automatique des règles de correction. Elle impose cependant à ce dernier de préciser les étapes de résolution du problème donné.

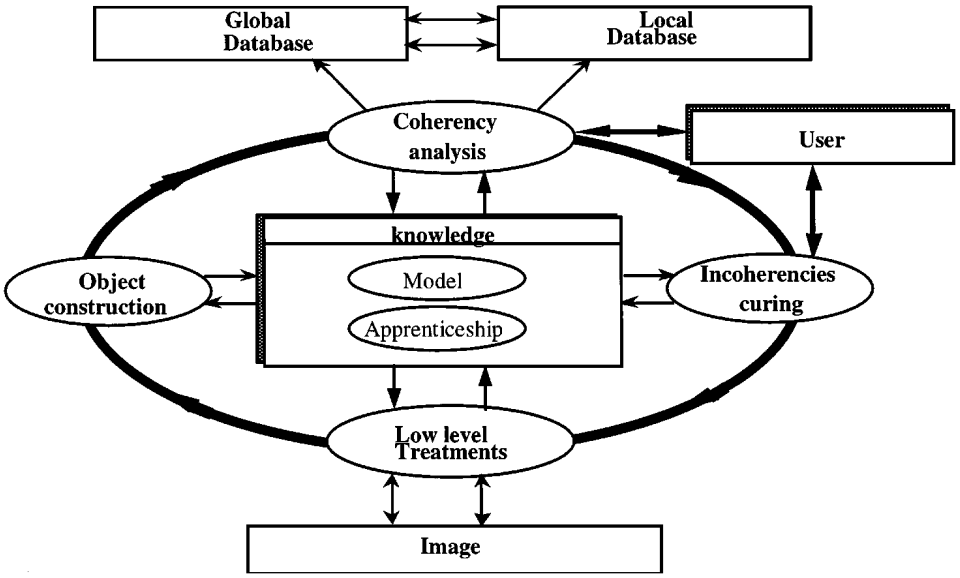


FIGURE 4.3 – Cycle de validation de la cohérence, selon l'approche d'Ogier *et al.* . (Extrait de [87, figure 15].)

Cette approche permet donc la mise en place progressive d'une connaissance algorithmique visant à gérer les cas particuliers des documents considérés. Toutefois, elle impose de spécifier précisément les opérations à réaliser pour replacer le système d'analyse dans un état correct, nécessitant alors une certaine expertise de l'humain sollicité et rendant délicat ce type d'interaction après une phase de mise au point du système.

En conclusion de cette section, on peut rappeler que si les approches visant à assister le système automatique en faisant intervenir ponctuellement l'humain semblent intéressantes pour minimiser l'effort manuel nécessaire en correction, cela présuppose une détection d'erreur très performante, et une exploitation maximale de l'information fournie par un groupe d'opérateurs humains infaillibles.

Dans le cas idéal, cela permet de ne réclamer que l'information la plus utile, et d'en tirer complètement profit en corrigeant les erreurs dans les résultats intermédiaires, puis dans les résultats produits, et il serait alors possible de disposer d'un nouvel exemple de page traitée pour permettre une adaptation du système.

En réalité, les systèmes actuels tirent peu profit de l'information apportée pour corriger les résultats intermédiaires erronés, car ils ne disposent pas d'un modèle de document leur permettant de réintégrer les données fournies par l'opérateur humain et de les fusionner avec celles extraites de l'image. La correction des erreurs, quant à elle, bénéficie rarement d'une interface permettant la remise en contexte de l'opérateur humain, et la validation des éléments saisis.

4.3 Conclusion

Au terme de ce chapitre, nous avons pu apprécier les différences dans les approches selon qu'elles laissent l'initiative à l'humain (assisté par la machine) ou au système automatique (assisté par l'humain).

Dans le cas des approches où les traitements sont *initiés par l'humain*, l'objectif est généralement de limiter de travail de conception, tout en faisant face à des données relativement complexes auxquelles l'humain s'adapte bien. Ceci permet d'envisager des résultats d'une qualité assez élevée, mais impose un coût de fonctionnement élevé.

À l'opposé, les approches où les traitements sont *initiés par la machine* s'attaquent différemment au problème, en cherchant à minimiser l'implication de l'humain dans le traitement, en particulier au niveau des corrections. Il s'agit dans ce cas d'orienter l'humain vers la correction ponctuelle de problème spécifiques qui doivent avoir été détectés au préalable. Ces systèmes peuvent atteindre des niveaux de qualité élevés si leur conception a été assez précise pour gérer efficacement les cas problématiques. Dans le cas de documents difficiles, le coût en correction est fortement lié au coût en conception : corriger efficacement les problèmes nécessite un travail important pour les anticiper.

Quel que soit le type d'approche considéré, face à des documents anciens et dégradés la production d'interprétations correctes est coûteuse, limitant alors fortement les possibilités d'amélioration du système à l'usage, à l'aide de mécanismes d'apprentissage. Il est donc nécessaire de permettre de tirer profit simultanément des informations extraites du fonds documentaire et de celles fournies par des opérateurs humains, pour pouvoir faire progresser à moindre coût le processus global, et espérer à terme intégrer des mécanismes d'amélioration.

Avant de finaliser le cahier des charges que nous avons débuté dans les conclusions des chapitres précédents, nous souhaitons nous intéresser brièvement à la variété des tâches que les approches existantes confient à l'humain, qui a pour effet de donner à ce dernier des rôles relativement distincts qu'on propose de définir précisément ici.

Oracle Il reçoit un élément à identifier, et propose une interprétation. Plus rarement, sa réponse au problème peut permettre de guider le système.

Guide Il fournit des informations au système qui lui permettent de progresser dans son analyse.

Expert documentaire Il détermine si une solution est correcte, et détecte les erreurs.

Superviseur Il supervise le fonctionnement du système.

Pilote Il choisit les traitements à réaliser sur les données.

Entraîneur Il gère le transfert de connaissance au système et son amélioration éventuelle.

Concepteur Il règle la mise au point du système en ajustant ses paramètres, ou en modifiant plus en profondeur ses modèles et son fonctionnement.

Nous nous focalisons dans nos travaux sur les rôles d'oracle, de guide et d'expert qui sont directement impliqués dans le travail d'interprétation sans nécessiter une connaissance technique du système, ce qui semble approprié dans un contexte de production où les experts techniques ne peuvent pas prendre part à l'interprétation des données. On parlera, ici et pour la suite, d'*opérateur humain* pour englober ces différents rôles.

Pour terminer ce chapitre, nous proposons un cahier des charges complet des fonctionnalités et des éléments nécessaires pour, partant d'un module d'interprétation de page isolée de référence, tel que décrit en conclusion du chapitre 2 (page 37), disposer d'un système d'interprétation contextuelle et assistée de fonds documentaire. Les points présentés ici reprennent et complètent les notes de la conclusion du chapitre 3 (page 49).

Mécanisme itératif pour une interprétation contextuelle et assistée Tout d'abord, il faut remarquer que l'interaction avec un opérateur humain peut être vue exactement de la même façon que l'interaction entre le module d'interprétation de page (au niveau de la page) et un niveau plus global chargé de tirer profit de contexte documentaire. Un *mécanisme d'interprétation itératif*, permettant un apport incrémental d'information, apparaît donc nécessaire, et doit permettre la mise en place d'une interprétation à la fois contextuelle et assistée, ces deux axes étant complémentaires. Ce mécanisme nécessite de pouvoir identifier, lors de la conception, les informations qui peuvent être échangées entre le module d'interprétation de page et un opérateur humain, et, comme lorsqu'on souhaite exploiter le contexte documentaire, déterminer quelles sont les contraintes entre les éléments extraits. La notion d'*environnement*, formé par les autres pages du fonds et par l'opérateur humain, peut donc être vue de façon homogène pour le module d'interprétation de page.

Niveau global Le niveau d'interprétation global, commun à toutes les pages du fonds documentaire, doit alors :

- centraliser les connaissances ;
- gérer automatiquement la circulation d'informations nécessaires à l'exploitation du contexte documentaire ;
- *gérer automatiquement la circulation d'informations nécessaires à l'interaction avec un ou plusieurs opérateurs humain.*

Interaction asynchrone L'interaction avec un opérateur humain impose la gestion d'une temporalité. Cette interaction doit être *asynchrone*, afin d'éviter de mettre le système ou l'opérateur humain en situation d'attente. La contrainte apportée par l'opérateur humain impose finalement une discipline qui est également nécessaire lorsqu'on souhaite tirer profit du contexte documentaire dans un fonds très volumineux : les dépendances entre pages et la nécessité de collecter des informations dans plusieurs pages pourraient provoquer des interblocages qu'il est alors possible d'éviter. Une base de donnée est nécessaire pour stocker les informations avant leur utilisation.

Fusion d'information et utilisation maximale Il est nécessaire, pour tirer profit d'informations contextuelles dans le module d'interprétation de page, de disposer d'un modèle de document qui permette la *fusion d'informations* entre les données extraites de l'image, et celles fournies par l'environnement : contexte documentaire ou opérateur humain, via un niveau plus global dans le système. Une approche efficace devra alors permettre de proposer une nouvelle solution optimale, en fonction de toutes ces données, de façon à *maximiser* l'utilisation des informations fournies par un humain. De façon homogène, un tel mécanisme peut être utilisé pour les informations extraites du contexte documentaire. Ceci impose de mettre en place des mécanismes de *remise en cause* des résultats.

Deux modes d'interaction complémentaires pour minimiser le travail manuel Donner un impact maximal à l'information fournie par l'environnement du module d'interprétation de page n'est pas nécessaire, il faut aussi s'assurer que cette information lui est utile. Il est donc nécessaire de mettre en place des mécanismes permettant, à partir des contraintes spécifiées par le concepteur du système, de *détecter automatiquement les problèmes* lors de l'interprétation, et de *réclamer la connaissance manquante.* Ceci doit permettre de minimiser la sollicitation de l'opérateur humain. On parlera alors d'*interaction dirigée*. Toutefois,

la détection automatique de problèmes n'étant pas toujours possible, il est important de laisser la possibilité à l'opérateur humain de détecter lui-même d'éventuels problèmes et de les corriger en fournissant *spontanément* de nouvelles informations. On parlera alors d'*interaction spontanée*. Ces deux modes doivent permettre de gérer dynamiquement la quantité d'information demandée à un opérateur humain.

Conception simple et centralisée Finalement les outils proposés doivent permettre de systématiser l'échange d'information entre le module d'interprétation de page et son environnement, afin de faciliter la production de résultats corrects à moindre coût, et sans augmenter la complexité de la conception du système, que ce soit au niveau de la page, ou au niveau global. Il faut donc permettre au concepteur du système d'exprimer de façon aussi *naturelle* et *centralisée* que possible sa connaissance à priori qui permettra de gérer automatiquement la mise en place de circulations complexes d'information. En particulier, il ne doit pas, au niveau de l'expression du modèle de la page, avoir à se préoccuper de contraintes temporelles. De la même manière, les éventuelles contraintes nécessaires à la saisie de données dans les interfaces homme–machine doivent être *transmises automatiquement* depuis le module d'interprétation de page, afin d'éviter toute duplication de la connaissance.

Maintenant que nous avons identifié *quels* éléments proposer et *pourquoi*, la prochaine partie va s'intéresser à *comment* les mettre en œuvre.

Deuxième partie

— Contribution —
Interagir avec l'environnement grâce à une interprétation itérative

Chapitre 5

Architecture globale pour une interprétation itérative

Introduction

Cette seconde partie présente notre contribution à l'interprétation de fonds documentaires anciens ou dégradés, c'est à dire de documents structurés, incertains, organisés en des ensembles finis et cohérents qu'il n'est pas nécessaire de traiter en flot continu. Elle est basée sur la proposition d'une architecture permettant l'*interprétation itérative* de pages, autorisant un apport *incrémental* et *asynchrone* d'informations contextuelles lors du traitement d'une page, et une progression de l'interprétation de cette dernière à chaque étape du processus. Ces informations peuvent provenir du fonds documentaire, c'est à dire d'autres pages, ou être avoir été saisies par des opérateurs humains, permettant donc la mise en place d'une interprétation à la fois *contextuelle* et *assistée*.

Dans les deux cas, la conception d'un module d'interprétation de page est peu perturbée et reste centrée sur la description d'une page, en vue d'un travail image par image, sans nécessiter la gestion des échanges entre ce module et son environnement (fonds documentaire et opérateurs humains). Les mécanismes que nous proposons permettent, par ailleurs, une diminution de l'effort de correction grâce à la fusion automatique des informations fournies par l'environnement et celles extraites de l'image lors de l'interprétation d'une page. Ceci permet la remise en cause automatique des anciens résultats et la production de nouveaux, basés sur des informations externes plus fiables. Il est également possible de mettre en place, au niveau du modèle de page, des mécanismes de détection d'erreurs réclamant des connaissances manquantes, et d'accepter ponctuellement le guidage de l'interprétation par son environnement.

Afin de permettre l'exploitation de propriétés globales du fonds documentaire, et gérer la circulation d'information entre les différents parties du système, un niveau global permet au concepteur de spécifier en un point unique la stratégie d'interprétation du fonds, qui peut considérer l'ensemble des données extraites à tout moment. Il est également possible, à ce niveau, de gérer automatiquement la quantité de travail manuel envoyé aux opérateurs humains, afin de maîtriser le rapport entre le coût de fonctionnement du système et la qualité des résultats produits.

Pour terminer cette introduction de notre contribution, nous insistons sur le point qui nous apparaît comme le plus original : pour traiter toutes les pages d'un fonds documentaire, le système que nous proposons pourra interpréter plusieurs fois chacune d'entre elles,

afin d'extraire progressivement toute l'information possible, en apportant de façon incrémentale la connaissance nécessaire, provenant d'autres pages ou d'opérateurs humains, à cette interprétation. Chaque page est *intégralement réinterprétée* à chaque itération, et les pages peuvent être traitées en parallèle au niveau global.

Le cadre que nous proposons, formé d'une architecture, mais aussi de méthodes pour le résolution d'un problème de traitement de fonds documentaire, donne une structure générique qu'un concepteur pourra utiliser et spécialiser à un fonds documentaire précis, afin de construire un système dédié à un cas particulier. La figure 5.1 illustre cette démarche de réutilisation et d'application de nos travaux.

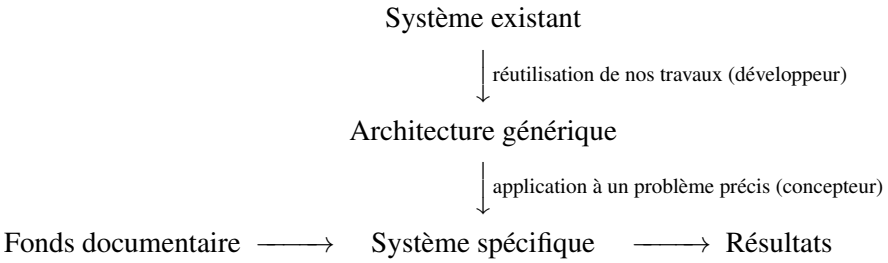


FIGURE 5.1 – Schéma de réutilisation de nos travaux pour la résolution d'un problème particulier.

Le chapitre 5 (le chapitre courant) présente l'architecture générale du système que nous proposons, et son comportement. Le module d'interprétation de page que nous avons identifié comme élément central de l'interprétation d'images de documents dans la partie I devient alors un des éléments essentiels de cette architecture.

La première section décrira les composants de l'architecture proposée, le rôle de chacun de ces éléments, et le travail restant à la charge du concepteur pour leur exploitation dans le cadre d'un problème particulier.

La seconde section montrera comment, en pratique, le système se comporte. Nous nous intéresserons aux modes d'interaction possibles avec un opérateur humain, avant de montrer comment ces modes peuvent être avantageusement utilisés pour mettre en place une interaction entre la page et son environnement en général. Il est en effet possible de gérer l'intégration d'information extraites d'autres pages exactement de la même manière que celles fournies par un opérateur humain, pour le concepteur du module d'interprétation de page.

Dans le chapitre 6 (le prochain chapitre), nous nous focaliserons sur l'identification précise des modifications à apporter à un module d'interprétation de page existant pour permettre son intégration dans un mécanisme d'interprétation itératif.

5.1 Composants nécessaires

Cette première section présente les composants de l'architecture du système d'interprétation contextuelle et assistée de fonds documentaires que nous proposons. Les principaux composants de ce système, et les flux d'informations entre ces derniers, sont illustrés dans

la figure 5.2. Le module d'interprétation de page devient alors un des élément essentiels de cette architecture globale, qui permet de lui apporter une connaissance supplémentaire grâce à un niveau d'abstraction dont il ne dispose pas. La construction de ce module, malgré les changements apparents, n'est que faiblement perturbée, et un élément existant pourra facilement être adapté pour s'intégrer dans cet environnement. Nous reviendrons plus en détail sur la façon de transformer un module (d'interprétation de page) de référence dans le prochain chapitre.

Nous nous intéressons ici à donner un niveau de détail suffisant pour reproduire nos propositions, mais laissons volontairement de côté un certain nombre d'éléments techniques qui perturberaient nos explications. Notons également que nous avons précisé, lorsque cela semblait nécessaire, le travail qui restait au concepteur d'un système spécifique lorsqu'il chercher à utiliser ces architectures génériques. En particulier, le concepteur sera responsable de la description du modèle de page, et de la mise en place de mécanismes de détection d'erreurs à ce niveau. À un niveau plus global, il devra exprimer comment tirer profit du contexte documentaire et gérer l'échange d'informations avec des opérateurs humains.

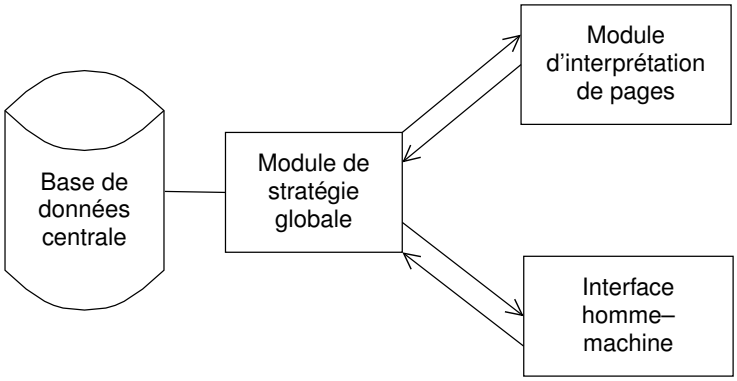


FIGURE 5.2 – Principaux composants d'un système d'interprétation contextuelle et assistée de fonds documentaires.

5.1.1 Base de données centrale

Le système est organisé autour d'une base de données qui contient une connaissance globale, qui croît progressivement au cours de l'interprétation, à propos du fonds documentaire. Cette base autorise l'exploitation de propriétés globales nécessitant de considérer l'information extraite de plusieurs pages afin d'améliorer la fiabilité de certains résultats ; et le regroupement d'éléments similaires avant de soumettre ces derniers à un opérateur humain. Elle permet également une communication asynchrone entre les différentes parties du système.

5.1.1.1 Données stockées et utilisation à l'exécution

Les données stockées par cette base de données centrale peuvent être de deux types.

Informations spécifiques à une image Pour ces informations, une localisation précise au sein de l'image est possible. Un exemple de telle donnée peut être l'emplacement de la colonne *« numéro de vente »* dans une image représentant un tableau ; ou encore

la transcription d'un mot précisément localisé. Ces données sont produites grâce à l'interprétation du contenu des images.

Informations relatives à un ensemble d'images Ces informations disposent d'une portée documentaire pouvant aussi bien concerner une seule image (par exemple le type de page concerné), quelques images (par exemple la taille des cadres d'un lot de formulaire précis), ou le fonds documentaire entier (par exemple le lexique des patronymes connus). Ces données peuvent être fournies au système avant son lancement sur un lot, ou acquises en cours de traitement, et peuvent évoluer en fonction des contenus détectés dans les images interprétées.

La nécessité de pouvoir reproduire l'organisation du fonds documentaire afin de tirer profit de ses propriétés, ainsi que la grande variabilité des données qui peuvent être stockées, imposent l'utilisation d'un schéma peu contraignant pour le concepteur. Ce dernier doit en effet, lors de la phase de conception du système, définir quelles seront les données à stocker et à manipuler au niveau global du système. Ces choix dépendent du fonds documentaire traité, et de l'approche choisie. Il faut dont permettre un stockage et un échange automatique des informations précédemment décrites entre les modules du système, tout en laissant une grande liberté au concepteur.

Lors de l'exécution, l'accès aux données de cette base peut être fait de deux façons :

1. en fonction des images concernées (portée documentaire) : c'est typiquement le cas lorsque qu'on cherche à collecter les informations à propos d'une image à traiter ;
2. en fonction de la nature des données (fonction logique) : c'est le cas lorsqu'on souhaite procéder à des regroupements de données similaires, par exemple tous les numéros de page d'un livre afin de valider leur cohérence, ou tous les patronymes d'un document afin de regrouper ceux qui sont visuellement similaires.

5.1.1.2 Représentation des données

De nombreux types de bases de données offrent la possibilité de stocker facilement ces informations, de façon relativement flexible, et permettent ces modes d'accès. Certains systèmes non-relationnels [1] rendent en effet possible un accès intéressant à un ensemble d'éléments de la forme suivante :

$$(\text{clé}_1, \text{clé}_2, \text{valeur}) \tag{5.1}$$

et autorisent des requêtes qui ne spécifient qu'un préfixe des clés pour accéder à l'information.

À l'aide d'un système de ce type, une représentation semblable à celle illustrée en table 5.1 peut être mise en place. Dans cet exemple, nous regroupons les données par première puis seconde clé pour permettre une représentation tabulaire. Ce type d'approche permet d'accéder facilement aux informations relatives à une page, ou à un ensemble de pages, et également de regrouper facilement des éléments selon leur type, comme par exemple des numéros de ventes ou des patronymes.

Dans l'exemple de la table 5.1, deux informations sont associées à l'élément *livre_1* : une information relative au lexique des patronymes utilisés, et une autre indiquant les paramètres du modèle utilisé pour reconnaître les tableaux contenus dans les images. Ces

1. Nous reviendrons sur ce point dans le chapitre 7 consacré à la mise en œuvre effective du système proposé.

Entité documentaire	Géométrie	Données
...		
livre_1		lexique_patronymes = *pat1.dic* taille_cadres = *dim1.par*
livre_1/page_1		page_type = *page_vide*
livre_1/page_2		page_type = *registre_ventes*
livre_1/page_2/annot_1	zone/coor_1 = {150, 210} zone/coor_2 = {505, 1720}	colonne_no_ordre
livre_1/page_2/annot_2	zone/coor_1 = {165, 216} zone/coor_2 = {204, 470}	numero_vente/val_finale = *15* numero_vente/lst_hypo = [(*15*, 0.87), ...]
...		

TABLE 5.1 – Exemple de données contenues dans la base de connaissance centrale.

informations peuvent être mises à jour au cours du traitement, si nécessaire : par exemple, lorsqu'un nouveau patronyme est validé par un opérateur humain. Au niveau de l'élément *livre_1/page_2*, plusieurs informations sont stockées :

1. une information relative à la nature de la page, qui peut permettre de déterminer l'outil d'interprétation à utiliser ;
2. une information à propos d'une colonne contenant des numéros de ventes, qui est localisée dans l'image ;
3. une information à propos d'un numéro de vente, lui aussi localisé dans l'image, pour lequel on dispose d'une valeur finale, et de la liste des hypothèses de reconnaissance.

Une extension utile de cette représentation serait d'ajouter des métadonnées pour faciliter la gestion des données, afin d'indiquer, par exemple, quel module a produit une certaine donnée et quand, à quel lot appartient chaque image, etc. Par rapport à l'exemple de la table 5.1, cela consiste à ajouter de nouvelles colonnes à notre représentation.

Une autre extension pourrait être d'ajouter une information de dépendance entre les données calculées, pour permettre une remise en cause en cascade lors de la modification d'un élément. Cela ne nous a pas semblé nécessaire dans nos premiers travaux, car comme nous le verrons au chapitre prochain, nous nous basons sur le modèle de page pour remettre en cause les résultats au niveau de l'interprétation de l'image.

Pour finir, on peut noter que ce type de base se prête bien à l'indexation des contenus et pourrait éventuellement être utilisé pour l'étape de diffusion auprès des utilisateurs finaux. Ceci éviterait les problèmes de recopie entre bases, fréquents lors de la livraison des résultats du traitement.

5.1.2 Module d'interprétation de page

Le module d'interprétation de page est le composant de base du système d'interprétation. C'est lui qui permet de passer les informations contenues dans une image d'une représentation visuelle à une représentation informatique adaptée, et c'est donc à lui qui

faut apporter la connaissance supplémentaire contenue dans l'environnement d'interprétation. Ce nouveau module d'interprétation de page travaille donc toujours page par page, mais il est capable de tirer profit d'information provenant de son environnement.

Dans ce chapitre, nous nous intéressons à la structure générale de l'architecture du système avant de rentrer dans les détails de la conception de ce module, qui fera l'objet du chapitre suivant. Par conséquent, nous nous intéressons ici à la façon d'intégrer ce module dans un système global, et donc au comportement qu'il doit avoir et aux hypothèses qu'on peut faire à son égard.

On peut toutefois préciser dès à présent que, concernant ce module, le rôle du concepteur reste inchangé : il s'agit pour ce dernier de décrire le contenu de l'image à l'aide d'un langage adapté, et, lors de l'exécution, ce module se chargera d'extraire automatiquement les contenus utiles dans l'image. À la différence de sa version classique, le module d'interprétation de page que nous proposons s'appuie sur un mécanisme d'interprétation itératif au niveau global. Ce module va alors gérer automatiquement l'échange d'informations avec son environnement, grâce aux nouvelles propriétés que le concepteur sera capable d'exprimer, pour faire progresser son interprétation. Ceci permet de simplifier le travail du concepteur, qui ne se préoccupe pas des contraintes temporelles, et n'a pas besoin de savoir comment les informations externes sont générées. Nous reviendrons en détail sur ces points dans le chapitre suivant.

Afin de faciliter la mise en place de ce mécanisme itératif, nous proposons d'utiliser une structure particulière pour servir de support à l'échange d'informations à propos des images des pages. Cette structure, que nous appelons « mémoire visuelle », vient s'ajouter aux éléments à fournir lors de l'invocation du module d'interprétation de page.

5.1.2.1 Mémoire visuelle

La *mémoire visuelle* est une structure associative particulière qui permet d'associer à chaque donnée des caractéristiques géométriques dans le référentiel de l'image à laquelle elle appartient. À titre d'exemple, le contenu de la mémoire visuelle associée à une page contenant un tableau de nombres pourrait être composé d'objets géométriques comportant des informations sur le type des colonnes, sur la position des nombres et leurs valeurs, sur la position des éléments suspectés d'être mal reconnus, etc.

On note $M_I(t)$ l'information contenue par la mémoire visuelle M associée à l'image I à l'instant t. Les données contenues sont du type suivant :

$$(\text{Zone} \times \text{TypeDonnée}) \rightarrow \text{Informations} \tag{5.2}$$

et elles peuvent donc être représentées par des triplets de la forme suivante :

$$(\texttt{zone}, \texttt{type}, \texttt{info}) \tag{5.3}$$

où `zone` représente la position et la forme de l'élément dans l'image courante, `type` représente un constructeur de type qui permet d'identifier la nature du contenu `info`. Selon le type de donnée, `info` peut contenir :

- *rien* : c'est le cas lorsque les caractéristiques géométriques et le type de la donnée suffisent à porter l'information, comme par exemple pour indiquer où se trouve la colonne contenant des numéros d'ordres ;
- *une valeur* : c'est le cas lorsque l'objet est atomique, comme par exemple pour indiquer le type d'une page ;

– *un ensemble de couples* (*attribut*: *valeur)* : c'est cas lorsque l'objet est structuré, comme par exemple pour décrire la valeur finale et la liste d'hypothèses d'un nombre.

Dans le cas de l'exemple de la table 5.1, la mémoire visuelle associée à une image représentant un tableau de ventes pourra contenir des éléments de la forme suivante :

$$
\begin{aligned}
M_{livre\_1/page\_2} = \{\ & (\textit{\_page\_entiere}, \texttt{lexique\_patronymes}, \texttt{pat1.dic}),\\
& (\textit{\_page\_entiere}, \texttt{taille\_cadres}, \texttt{dim1.par}),\\
& (\textit{\_page\_entiere}, \texttt{page\_type}, \texttt{registre\_ventes}),\\
& ([(150, 210), (505, 1720)], \texttt{colonne\_no\_ordre}, \_),\\
& ([(165, 216), (204, 470)], \texttt{numero\_vente},\\
& \qquad \{\ \texttt{val\_finale}: 15,\\
& \qquad\ \ \texttt{lst\_hypo}: [(15, 0.87), \ldots]\ \})\ \}
\end{aligned}
$$

La mémoire visuelle associée à une image peut donc être générée automatiquement en fonction du contenu de la base de données centrale.

Comme nous le verrons dans le prochain chapitre, cette structure permet de simuler une structure de données persistante que le module d'interprétation de page utilise pour reprendre son travail d'interprétation lors de chaque itération. Son exploitation au cœur de l'interprétation d'image en fait la *structure pivot* permettant de fusionner l'information provenant de l'environnement et celle contenue dans l'image.

La mémoire visuelle est une structure efficace pour permettre la réintégration d'informations relatives à une image donnée, mais il peut être nécessaire d'apporter au module d'interprétation de page d'autres informations, comme une référence à un lexique par exemple, héritées de l'ensemble documentaire plus large auquel la page en cours d'analyse appartient. Dans ce cas, on peut ajouter de nouveaux canaux de communication pour transmettre cette information, ou se servir de la mémoire visuelle en donnant à l'information une portée couvrant toute la zone de l'image (comme illustré dans l'exemple précédent), pour indiquer que cette information est valide pour la surface complète.

5.1.2.2 Comportement du module d'interprétation de page

Intégré dans l'architecture globale du système, le module d'interprétation de page travaille à faire évoluer la connaissance relative aux images pour lesquelles il est sollicité. Grâce à un modèle de document spécifique, le module d'interprétation de page traite à la demande une image I accompagnée de la mémoire visuelle associée $M_I(t)$. En retour, il propose une nouvelle version de l'information relative à l'image $M_I(t+1)$, et éventuellement d'autres résultats (l'information finale n'est pas obligatoirement contenue dans la mémoire visuelle) et codes d'erreurs. D'un point de vue abstrait, le module d'interprétation de page a une signature de la forme suivante :

$$(\text{Image}, \text{Mémoire}) \rightarrow (\text{Résultat}, \text{Mémoire}) \tag{5.4}$$

Si le module d'interprétation de page n'arrive pas à interpréter certains éléments, alors il peut stopper son traitement et stocker des questions dans la mémoire visuelle de sortie pour indiquer les problèmes qu'il a rencontrés, les options possibles ou les cas testés, ou encore des informations contextuelles pour faciliter une correction manuelle éventuelle.

La seule contrainte forte imposée à ce module est la nécessité de tirer profit de l'information externe, si elle est satisfaisante, pour faire progresser l'interprétation de l'image

courante, et viser une terminaison du traitement. En particulier, si une réponse satisfaisante a été apportée suite à une question posée par le module, ce dernier doit utiliser la réponse pour progresser et ne pas reposer la même question. Nous verrons dans le chapitre suivant comment de telles propriétés peuvent être, dans une certaine mesure, garanties. Le module d'interprétation de page se comporte alors comme une des étapes d'un mécanisme itératif global qui ressemble, s'il est naïf, à un algorithme glouton.

La nécessité pour le module de pouvoir s'arrêter, puis reprendre son travail, à des conséquences non négligeables dans le cadre de traitement de données en volume important, car cela impose de terminer le processus lié et de libérer ses ressources, puis de le relancer lorsque c'est nécessaire. Il n'est en effet pas possible de conserver dans la mémoire de la machine l'intégralité des processus en attente d'informations contextuelles, lors du traitement d'un très grand nombre de pages. Ceci est renforcé par le besoin de mettre en place une interaction asynchrone entre les différentes parties du système. Par ailleurs, la nécessité pour le module d'interprétation de page de pouvoir remettre en cause l'interprétation qu'il a faite d'une image à une itération précédente impose, dans une certaine mesure, de réinterpréter les données. Nous verrons dans le chapitre prochain comment limiter la charge supplémentaire imposée par cette approche en conservant des résultats intermédiaires.

5.1.3 Module de stratégie globale

Le module de stratégie globale est le « cerveau » du système d'interprétation de fonds documentaires. « Globale » indique ici qu'il a une connaissance de l'intégralité de l'information relative au fonds documentaire à un instant donné, par opposition aux modules de traitement qui ont une vision limitée de la connaissance et des données. Parmi ces modules focalisés sur une partie des données, on peut distinguer :

- ceux travaillant au niveau d'une page, comme le module d'interprétation de page et certaines interfaces de visualisation et de correction ;
- et ceux travaillant à un niveau plus abstrait, fait d'ensemble d'annotations par exemple, comme un éventuel module de clustering de vignettes de mots, et les interfaces de visualisation associées.

Le rôle du module de stratégie globale peut être vu selon deux angles : au plus près de la gestion des donnés, il est chargé de coordonner et ordonnancer les différents modules du système, tandis qu'à un niveau plus abstrait, il permet de tirer profit des propriétés du fonds documentaire tout en interagissant avec les opérateurs humains. Nous détaillons ci-après ces deux aspects, avant de nous intéresser brièvement au travail de conception nécessaire au niveau de ce module.

5.1.3.1 Médiation et coordination des modules

Le rôle de médiateur du module de stratégie globale est lié à la nécessité de conserver une cohérence dans l'information stockée dans la base de données centrale, et limiter la connaissance requise par chaque composant du système afin de faciliter leur mise en œuvre. Ceci permet, d'un point de vue pratique, de garantir une bonne circulation de l'information entre les composants. En outre, cela permet de contrôler les modifications faites au niveau de la connaissance globale, ce qui permet de limiter l'impact d'erreurs des traitements automatiques et d'opérateurs humains. Le module de stratégie globale est alors responsable du choix de fusionner, supprimer ou transmettre les informations produites, et de les enregistrer si nécessaire.

Ce rôle de médiation est complété d'un rôle d'ordonnanceur qui rend le module responsable de l'invocation ou de la sollicitation des modules et interfaces utiles au cours du déroulement d'un scénario global. Sa capacité à utiliser les données produites lui permet, dans une certaine mesure, d'adapter l'enchaînement des opérations à effectuer.

5.1.3.2 Interprétation globale contextuelle et assistée

À un niveau plus abstrait, le module de stratégie globale peut avoir plusieurs fonctions que nous détaillons dans les paragraphes suivants.

Gestion de la variabilité des types de pages Pour faire face à la variation de types de pages (ex : couverture, sommaire, texte), il est possible de décider au niveau de ce module des opérations à réaliser si l'information est présente dans la base centrale. Si ce n'est pas le cas, il est possible d'exploiter des modules de détection (classification de type de document), ou de tirer profit de certaines règles de composition du fonds documentaire considéré : dans le cas d'un livre, par exemple, on peut indiquer que la couverture est suivie de quelques pages de nature variable, puis d'un sommaire d'une ou plusieurs pages, etc. Nos tentatives de description de cette variabilité nous ont montré qu'il ne semblait pas nécessaire de mettre en place un niveau de langage supplémentaire ; un automate régulier suffit généralement pour déterminer l'ensemble des types pour une page donnée. Cependant, ceci pourrait être envisageable au niveau du module de stratégie globale.

Exploitation du contexte documentaire – interaction avec le fonds Le niveau d'abstraction du module de stratégie globale permet d'exploiter les propriétés d'un fonds documentaire, en particulier les redondances dans les contenus et les liens sémantiques inter-pages. Ceci se fait en mettant en place un aller-retour d'informations visant à :

- collecter les interprétations locales (interprétation de page, typiquement) ;
- puis à les regrouper pour optimiser la réponse globale en validant des contraintes numériques ou de similarités entre les éléments (cas de mots visuellement similaires, ou de séquences) ;
- et à réinjecter cette information dans les modules d'interprétation pour les forcer à mettre à jour leurs résultats, si nécessaire.

D'autres formes de circulation de l'information sont également envisageables. Il serait par exemple possible de transmettre des informations d'une page à l'autre : en transmettant le numéro de la page courante à la page suivante, il serait possible d'améliorer la reconnaissance de ces numéros. Cela semble toutefois moins intéressant dans la mesure où le contexte utilisé est très restreint, et que cela alourdit la conception des traitements locaux.

Gestion des interactions avec les opérateurs humains Le niveau global de ce module est également le niveau approprié (car unique) pour gérer les interactions avec les opérateurs humains, soit en les sollicitant directement pour régler des problèmes détectés, soit en leur laissant la possibilité d'intervenir plus librement à certains moments du scénario considéré. Le module peut alors regrouper, filtrer ou trier les demandes faites par les modules automatiques pour présenter les éléments les plus pertinents aux opérateurs humains, en particulier grâce à la considération du contexte documentaire. Par ailleurs, le module permet de répartir le travail entre les différents opérateurs, si nécessaire, et de coordonner leur travail. Dans le cas de tâches particulièrement difficiles ou de petits lots, il est possible de n'utiliser que très peu de modules automatiques pour se concentrer sur une assistance minimale des

opérateurs humains, ce qui offre une grande flexibilité dans les modes de fonctionnement possibles sans alourdir le travail de conception.

Amélioration des modèles Même si nous ne présentons, dans ces travaux, aucune application de méthodes d'apprentissage, nous avons pris soin de rendre possible l'intégration de mécanismes d'amélioration et d'adaptation. Le module de stratégie globale peut donc, s'il dispose de tels outils, réaliser des apprentissages à partir des données correctement reconnues. Plus simplement, il est possible de ne modifier que certains paramètres des modèles, comme des lexique de mots ou des dimensions de cadres de formulaires, lorsque les informations sont reconnues comme fiables.

5.1.3.3 Rôle du concepteur

Malgré la variété des responsabilités du module de stratégie globale, le travail du concepteur de ce module n'est pas particulièrement complexe, à condition de disposer de librairies ou d'outils facilitant la définition d'enchaînements de traitements et l'invocation des modules concernés. Ce dernier doit être en mesure de déterminer quelles propriétés du fonds documentaire il souhaite exploiter (redondances, séquences, etc.) et quels sont les algorithmes ou modules lui permettant de le faire. Il doit également être capable de déterminer quand solliciter les interfaces homme–machine permettant de bénéficier de la connaissance des opérateurs humains.

Ces décisions donnent lieu à l'écriture d'un algorithme, ou d'un ensemble de règles, régissant l'invocation des différents modules en fonction des données produites par ces derniers, et de la connaissance contenue dans la base de données centrale. Dans la mesure du possible, la circulation de l'information entre les différents modules doit être gérée automatiquement par des outils dédiés. À l'usage, nous nous sommes aperçus qu'il était pratique de déterminer un ensemble d'états possibles pour les données à traiter (ex : page en attente de (ré)interprétation, page en attente d'interaction, etc.) et les transitions possibles entre ces états, pour générer automatiquement un algorithme de pilotage.

5.1.4 Interfaces homme – machine

Les interfaces de communication avec les opérateurs humains permettent la mise en place d'un échange entre ces derniers et le système d'interprétation, plus précisément le module de stratégie globale. Ces interfaces peuvent être de plusieurs types, selon la tâche qu'elles permettent de réaliser. On peut en effet les distinguer selon deux axes :

1. selon le niveau d'abstraction des données qu'elles permettent d'éditer : l'édition peut-être faite à l'échelle d'une page, ou à un niveau plus abstrait, pour valider et corriger des regroupements ;
2. selon le degré de liberté laissé à l'opérateur humain lors de la saisie des informations : cette saisie peut être guidée par les problèmes détectés, et ne fournir que le contexte nécessaire à l'opérateur pour lui permettre de prendre une décision rapide, ou alors laisser l'opérateur explorer librement certaines données et déterminer lui-même les informations qu'il doit fournir, comme par exemple pour détecter et corriger des cas de sous-segmentation dans une image de page.

Dans tous les cas, il est important de noter que les informations utilisées et produites par ces interfaces sont du même type que celles produites par les modules automatiques. Ceci

est une condition nécessaire pour permettre la mise en place d'une collaboration homme–machine au sein du système. Le modèle de représentation cible des données doit alors être défini clairement et respecté par les différentes parties, à la manière d'un contrat à propos de la forme des résultats à produire. Les interfaces doivent donc contenir une connaissance supplémentaire, spécifique à un type de document et fournie par le concepteur du système, permettant de présenter les données et de les éditer. Cette connaissance permet la transformation de données bien identifiées (par exemple les positions des cadres d'un formulaire, ou encore les nombres localisés et reconnus dans un tableau) dont les modifications permettront de guider, corriger ou valider l'interprétation automatique.

Afin de permettre de guider l'opérateur humain dans des choix (par exemple la valeur d'un nombre), accélérer sa saisie, ou limiter les risques d'erreurs, il est possible d'ajouter des contraintes de domaines, de position ou autres, aux données transmises à l'interface afin de limiter les éditions possibles. Il est également possible de n'autoriser que certains types de données à être modifiés, voire à en masquer certains, selon les besoins.

Finalement, du point du vue du système, et plus particulièrement du module d'interprétation de page, ces interfaces peuvent être considérées comme des groupes de ressources qui pourront être sollicitées pour des tâches précises, au cours d'un enchaînement d'opérations sur les documents et les données produites, à l'instar des modules de traitement automatique.

5.2 Utilisation du système global et comportement

Cette section montre comment le système global présenté dans la première section permet, au niveau du module d'interprétation de page, de tirer profit d'informations fournies par les opérateurs humains, mais aussi d'informations provenant du contexte documentaire. Nous verrons qu'au niveau de ce module, il n'est finalement pas nécessaire de distinguer la source de ces nouvelles connaissances, car le modèle d'interaction global que nous proposons est suffisamment général pour lui permettre de faire l'hypothèse, lors de la conception du modèle de page, que les connaissances manquantes seront fournies par l'environnement, sans se préoccuper de la manière dont elles seront produites.

5.2.1 Interaction avec des opérateurs humains

Deux modes d'interaction complémentaires peuvent être utilisés pour communiquer avec les opérateurs humains dans le système que nous proposons. La capacité à détecter les erreurs est le critère essentiel qui permet de décider du mode d'interaction à utiliser.

1. Le premier, celui à privilégier, est le mode *dirigé* (par la détection automatique d'erreurs) pour lequel, lorsque la détection d'erreur est possible, le système pose une question (à laquelle un opérateur humain doit répondre) à chaque fois qu'il détecte une erreur.
2. Le second, qui complète le premier, est le mode *spontané*, adapté aux situations dans lesquelles la détection automatique d'erreurs n'est pas possible.

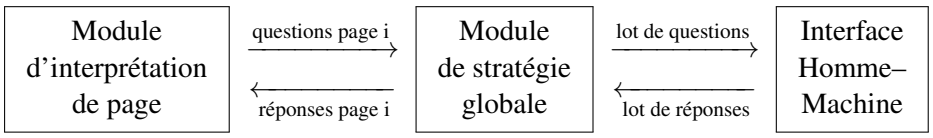
La mise en place du mécanisme de détection d'erreur dépend du concepteur du module d'interprétation de page (et du module de stratégie globale, mais on laissera ce cas de côté pour simplifier les explications). En effet, c'est grâce à l'expression de contraintes entre les éléments détectés au sein d'une ou plusieurs pages, et de mécanismes de rejet basé sur la confiance accordée à la localisation ou à la reconnaissance de ces éléments, qu'il est

possible de mettre en place un mécanisme automatique sollicitant un opérateur humain (ou d'autres modules automatiques) pour compléter la connaissance du système.

5.2.1.1 Interaction dirigée

L'interaction *dirigée*, c'est à dire imposant aux opérateurs humains la résolution d'un problème précis, est tributaire d'une détection relativement fine des erreurs, en particulier au niveau de leur localisation, afin de permettre de *poser une question* faisant référence à la *zone concernée* qui permet de décrire le *contexte visuel minimal* nécessaire pour répondre. Ce mode est très similaire à ce que Nagy et Veeramachaneni [82] appellent les interactions « initiées par la machine », où l'information à traiter par les humains est déterminée automatiquement.

Vue globale des informations échangées Comme illustré ci-dessous, le module d'interprétation de page produit automatiquement des questions à destination de son environnement, sans savoir comment les réponses seront produites. Ces questions sont générées suite à la détection d'incohérences, ou de manque de connaissances pour traiter une situation, grâce à des mécanismes de contrôle définis par le concepteur du système. Le module de stratégie globale collecte ces questions et lorsqu'il est possible de constituer un lot de questions de taille suffisante pour mobiliser un opérateur humain, ce lot est transmis à l'interface permettant de résoudre les problèmes repérés.



Représentation des questions et des réponses Les questions contiennent les informations suivantes :

- la localisation du problème ;
- d'autres éléments de contexte permettant de mieux comprendre la raison de la question : structure en cours d'analyse, éléments préalablement détectés, choix automatiques précédant l'erreur, ou plus simplement une phrase générée dynamiquement ;
- des contraintes ou suggestions pour les réponses possibles : type de données attendues, restriction sur le domaine des valeurs possibles, positionnement et forme de la réponse, liste d'hypothèses ambiguës, etc.

Les types des données qui peuvent être réclamées dans les questions sont prédéfinis par le concepteur (ex : valeur d'un nombre, position d'une colonne, etc.) selon le fonds traité et l'approche choisie. Ces questions sont instanciées (automatiquement) sur des éléments précis à l'exécution.

Les questions sont représentées par des données en mémoire visuelle de la forme suivante (le format correspondant en base est équivalent, à une transformation près) :

$$(\text{zoneQuestion}, \texttt{question}, \{\ \text{contexte}, \text{explications}, \text{contraintesReponse}, \ldots\ \})$$

les réponses peuvent, quant à elles, être représentées de la façon suivante dans la mémoire visuelle.

$$(\text{zoneReponse}, \text{typeReponse}, \text{contenuReponse})$$

Nous préciserons ces formes, encore générales, dans la sous-section 6.3.1 du prochain chapitre.

Actions de l'opérateur humain Les questions étant spécifiques à un type de problème pré-défini, il peut être judicieux de les regrouper selon ces types pour permettre l'envoi de lots homogènes à des interfaces qui peuvent alors être spécialisées, comme par exemple pour le vidéo-codage de zones de texte incertaines. À la réception de ces questions, l'opérateur humain utilise les outils à sa disposition pour les visualiser et y répond progressivement. Dans ce mode, l'opérateur *doit* apporter une réponse pour *chaque* question. Lorsqu'il a terminé, il valide ses réponses et les soumet au système qui aura, si possible, traité d'autres données entre-temps, et généré un nouveau lot de questions.

Stratégie globale Au niveau du module de stratégie globale, il est nécessaire de pouvoir gérer l'évolution de l'état des pages en fonction des questions et réponses contenues dans la mémoire visuelles associées à ces dernières. La façon la plus simple de gérer le traitement d'un ensemble de pages selon ce mode *dirigé* consiste à créer un automate qui sera suivi par chacune des pages traitées en parallèle. La figure 5.3 donne un exemple d'automate pour ce mode d'interaction.

Cet automate met en attente de traitement automatique chaque nouvelle page. Lorsqu'une page a été interprétée par le module d'interprétation de page, elle est mise en attente de traitement manuel si ce dernier a généré des questions. Sinon, elle est marquée comme traitée. Lorsque les réponses aux questions associées à une page sont toutes disponibles, cette dernière est à nouveau mise en attente de traitement automatique.

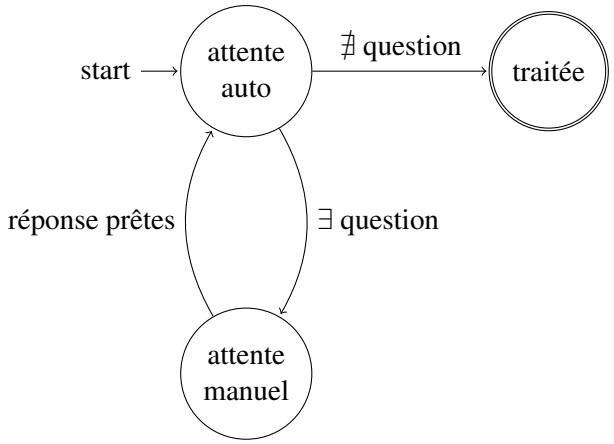


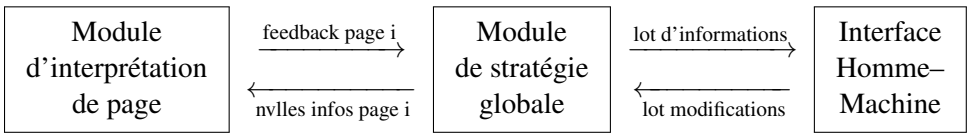
FIGURE 5.3 – Exemple d'automate définissant les états possibles d'une page dans un mode d'interaction dirigée. C'est une forme de spécification possible de la stratégie d'interprétation globale par le concepteur.

Ce mode d'interaction est particulièrement pertinent lorsque la détection d'erreurs automatique est efficace, car il permet de réclamer les informations les plus utiles pour le système, et de limiter le travail des opérateurs humains au strict nécessaire. Toutefois, cela n'est pas toujours possible, et nous avons eu besoin de recourir à un second mode de fonctionnement dans certaines situations.

5.2.1.2 Interaction spontanée

Ce second mode vise à proposer une alternative à la nécessité de détecter automatiquement les problèmes en cours de traitement. Nous appelons *spontané* ce second mode d'interaction, car il laisse, dans une certaine mesure, les opérateurs humains déterminer eux-mêmes les informations qu'ils fournissent au système. Ceci permet d'autoriser la détection d'erreurs *par les opérateurs humains*. Contrairement au mode *dirigé*, le mode *spontané* ne peut pas être mis directement en correspondance avec le second mode d'interaction identifié par Nagy et Veeramachaneni [82], celui où les interactions sont « initiées par l'humain ». En effet, dans notre cas, le système ne réagit pas directement aux informations saisies par les opérateurs pour les interpréter, mais il autorise, à certaines étapes de son fonctionnement, l'apport d'information externe de types pré-définis, qui compléteront ou corrigeront des informations déjà produites par le système.

Vue globale des informations échangées Comme pour le mode d'interaction précédent, c'est le module d'interprétation de page qui est à l'origine de l'information autorisant l'action d'opérateurs humains. Dans ce mode, le module d'interprétation de page produit, pour chaque image, un compte-rendu, ou plus précisément un ensemble d'indices, montrant les résultats qu'il a produits ainsi que certains résultats intermédiaires, si nécessaire, et des indications sur les apports d'information possibles. Le module de stratégie globale collecte ces données, et permet, par lots, des modifications ou des ajouts d'informations pour chaque page. Grâce à ces nouvelles informations, il est possible de relancer le module de stratégie globale sur les pages concernées pour faire progresser leur interprétation. Ce mécanisme est illustré ci-dessous.



Représentation des données Les informations renvoyées par le module d'interprétation de page sont de deux types :

1. des résultats produits par le module, représentées en mémoire visuelle par des données simples ;
2. des indications sur les informations qu'il est possible de fournir, représentées en mémoire visuelle par des questions implicites et optionnelles.

Les « questions optionnelles » décrivent, pour une zone de l'image donnée, des contraintes quant aux types de données qu'il est possible d'ajouter ou modifier, ainsi que le nombre d'éléments qu'il est possible d'ajouter. Elle sont représentées par des données en mémoire visuelle de la forme suivante.

$$(\text{zoneQuestion}, \texttt{question\_optionnelle}, \{\ \text{contexte}, \text{explications}, \text{contraintesRep}, \ldots\ \})$$

les résultats renvoyés par le module d'interprétation de page et les réponses fournies par l'opérateur humain prennent la même forme générale et sont homogènes :

$$(\text{zone}, \text{type}, \text{contenu})$$

C'est au concepteur du module d'interprétation de page de déterminer quelles informations pourront être renvoyées pour indication, et quelles seront les informations qui pourront

être apportées par l'opérateur humain. Par exemple, il est possible d'indiquer que dans des images contenant des tableaux dont la position des colonnes est incertaine, l'opérateur humain aura le droit de déplacer la position de ces colonnes pour corriger spontanément d'éventuelles erreurs de localisation. Dans la détection de nombres dans la colonne d'un tableau, on pourra autoriser l'ajout de nombres qui auraient été manqués.

Actions de l'opérateur humain Dans ce mode d'interaction, on a tendance à privilégier une visualisation image par image pour donner à l'opérateur humain la possibilité d'interpréter chaque élément en contexte. Les contraintes relatives aux informations qu'il peut apporter sont affichées dans l'interface, et il choisit le type, la position, et le contenu des données qu'il veut ajouter, en nombre libre, pour compléter l'interprétation de l'image courante. Il peut également modifier des résultats intermédiaires et, dans certains cas, supprimer des éléments erronés.

Stratégie globale Au niveau de la stratégie globale, ce mode d'interaction impose une sollicitation presque systématique de l'opérateur humain pour valider les résultats associés à une image. Bien entendu, dans des cas réels, une détection plus large du besoin de contrôle manuel est possible, mais nous nous attachons ici à décrire deux modes d'interaction complémentaires, qui seront utilisés simultanément en pratique. La façon la plus simple de gérer ce mode *spontané* vise, comme dans le cas précédent, à définir un automate comme celui de la figure 5.4 permettant le traitement simultané d'un grand nombre d'images.

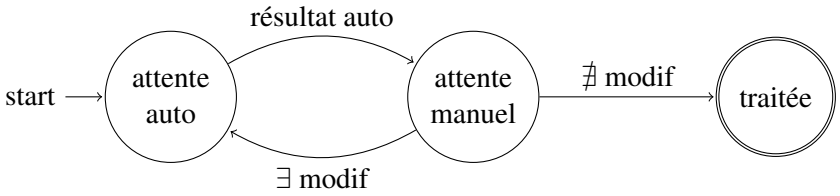


FIGURE 5.4 – Exemple d'automate définissant les états possibles d'une page dans un mode d'interaction spontanée. C'est une forme de spécification possible de la stratégie d'interprétation globale par le concepteur.

Bien qu'il soit *à priori* plus coûteux que le mode dirigée, ce mode a l'avantage majeur de ne pas dépendre d'une détection d'erreurs automatique précise, et permet éventuellement de corriger la cause du problème, comme par exemple la position de la colonne des numéros de ventes, entraînant potentiellement une remise en cause en profondeur de l'interprétation d'une image. Ce second mode est, en pratique, utilisé simultanément et en complément du premier. C'est alors au concepteur de déterminer lequel utiliser, en fonction des mécanismes de détection automatique de problèmes qui peuvent être mis en place. La mise en œuvre de ces deux modes sera présentée dans la sous-section 6.3.1 du prochain chapitre.

5.2.2 Interaction avec le fonds documentaire

L'exploitation du contexte documentaire, mise en œuvre à l'aide d'une interaction entre la page et son environnement, ou plus précisément entre le module d'interprétation de page et le module de stratégie globale, doit être construite en complémentarité avec l'interaction

entre le système et l'opérateur humain. Dans l'approche que nous proposons, il est possible de mettre en place ces deux formes générales d'interaction en se basant *exactement* sur les mêmes mécanismes, à savoir le mode *dirigé* et le mode *spontané* présentés précédemment.

Ceci permet une séparation claire des responsabilité du module d'interprétation de page et du module de stratégie globale, ce qui simplifie leur conception.

- Le module d'interprétation de page est basé sur une connaissance locale, exprimée par le concepteur, centrée sur la description de l'image et sur la production de résultats. Cette description exprime des contraintes entre les contenus qui peuvent permettre de détecter certaines erreurs, et permet la réintégration d'informations externes si elles existent. Elle ne fait *aucune hypothèse* sur la façon dont sont produites ces données externes par l'environnement du module d'interprétation de page, et ne gère pas la circulation des données.
- Le module de stratégie globale est quant à lui basé sur une connaissance globale, exprimée par le concepteur, lui permettant de répartir le travail entre les opérateurs humains et d'exploiter certaines propriétés du fonds documentaire. Cependant, cette connaissance ne décrit pas l'organisation interne de la page, ni les contraintes locales entre les éléments.

Pour illustrer cette séparation des responsabilités entre les différents niveaux du système que nous proposons, nous décrivons ci-après, brièvement, deux exemples de scénarios montrant qu'il est possible de gérer simplement plusieurs niveaux de remise en cause dans les résultats produits. Le premier exemple s'intéresse à la remise en cause d'éléments terminaux (cellules dans une structure tabulaire) ; et le second montre comment une remise en cause plus en profondeur est possible avec la modification d'éléments structurels (localisation d'une colonne dans une structure tabulaire).

Dans les deux cas, le contexte documentaire et la connaissance de l'opérateur humain sont utilisés simultanément pour apporter de nouvelles informations au module d'interprétation de page et faire progresser son interprétation pour les pages problématiques.

5.2.2.1 Optimisation d'une séquence de nombres et interaction spontanée

Dans cet exemple, nous essayons d'extraire la structure d'un lot de page (un registre) représentant des tableaux de ventes. Nous supposons ici que, dans ces tableaux, la reconnaissance des numéros de ventes (illustrés en figure 5.5), peut être peu fiable à cause du style d'écriture. Cependant, en sachant que ces nombres respectent un incrément de une unité entre chaque occurrence, et ce sur l'ensemble des pages d'un registre. Le scénario, dont la description succincte, suit permet de tirer profit de cette propriété, et de la disponibilité d'un opérateur humain, pour produire des résultats structurés fiables.

1. Pour chaque page du lot à traiter, le module de stratégie globale invoque le module d'interprétation de page, qui repère les numéros de ventes, produit une liste d'hypothèses pour chacun, et construit une structure résultat pour chaque ligne de vente (contenant d'autres éléments détectés). Le module d'interprétation de page renvoie au module de stratégie globale, conformément au protocole du mode d'interaction *spontanée*, la position et les hypothèses de reconnaissance pour chacun des nombres dans la mémoire visuelle associée à chaque image.
2. Le module de stratégie globale regroupe les hypothèses relatives aux numéros de ventes sur plusieurs pages, et détermine la valeur la plus probable de chaque élément en contexte à l'aide d'un algorithme comme celui de Viterbi.

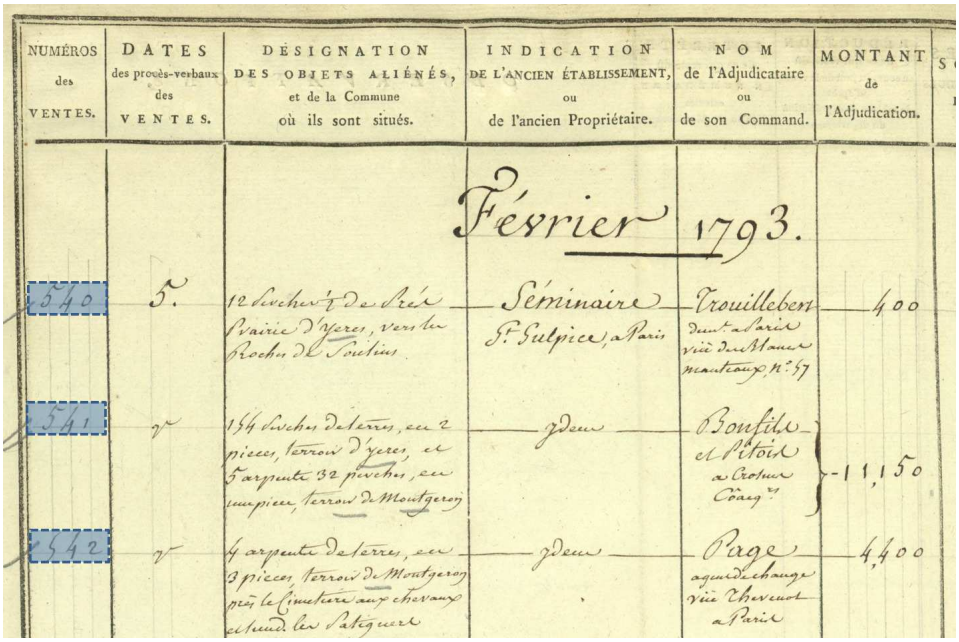


FIGURE 5.5 – Exemple extrait d'une page où l'on souhaite localiser puis reconnaître les numéros de ventes encadrés en bleu.

3. Si des éléments semblent poser problème (suspicion de saut, etc.) alors un opérateur humain peut être sollicité pour indiquer la valeur réelle des éléments.
4. Après une nouvelle optimisation avec ces informations fiables, le module de stratégie globale invoque à nouveau le module d'interprétation de page pour lui transmettre les valeurs finales des numéros de ventes.
5. Le module d'interprétation de page réintègre automatiquement ces informations, grâce au mécanisme itératif qu'il intègre (mémoire visuelle et description adaptée), pour générer une nouvelle structure résultat intégrant ces informations fiables.

Dans ce scénario, le concepteur du module d'interprétation de page n'a pas besoin de se préoccuper de savoir comment les valeurs des numéros de ventes pourront être fiabilisées grâce à l'environnement du module : il peut se concentrer sur la description du contenu. Symétriquement, le concepteur du module de stratégie globale se concentre exclusivement sur la circulation de l'information entre les modules, et l'exploitation d'une propriété globale du fonds. À aucun moment il n'a besoin de connaître la structure de la page. Les deux niveaux de conceptions sont alors quasiment indépendants, et leur réutilisation et amélioration peuvent être réalisées séparément.

5.2.2.2 Relocalisation d'une colonne dans un tableau et interaction dirigée

Dans cet exemple, nous essayons de localiser correctement la position de la colonne contenant les numéros de vente, préalable à l'extraction présentée dans l'exemple précédent. La figure figure 5.6 illustre la position de cette colonne sur un exemple. Nous supposons ici que la localisation de cette colonne peut être perturbée par l'effacement du cadre du tableau, utilisé comme indice pour le recalage, et qu'il est fréquent de tenter de reconnaître

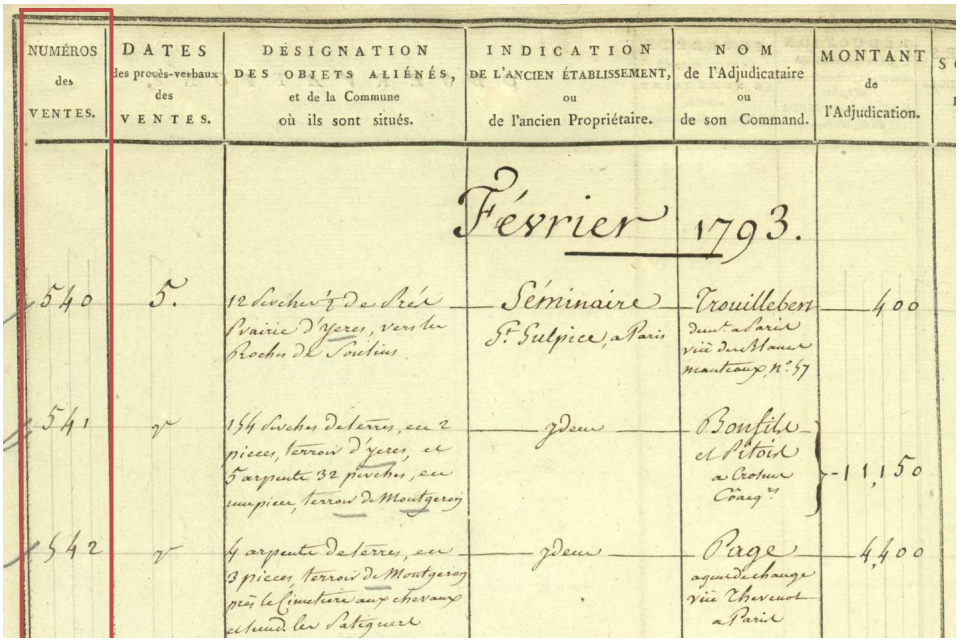


FIGURE 5.6 – Exemple extrait d'une page où l'on souhaite localiser la colonne encadrée en rouge, contenant les numéros de ventes.

les éléments de la deuxième colonne (en partant de la gauche) contenant des dates. En supposant qu'essayer de lire des numéros de vente dans la deuxième colonne produise de très mauvais résultats, il est possible de s'appuyer sur le scénario, dont la description succincte suit, pour corriger au plus tôt ce problème de localisation.

1. Pour chaque page du lot à traiter, le module de stratégie globale invoque le module d'interprétation de page qui tente de localiser la colonne des numéros de ventes et de reconnaître les éléments contenus. Si les éléments sont globalement peu fiables, le mécanisme de détection défini par le concepteur du module d'interprétation de page génère une question destiner à notifier de module de stratégie globale d'un manque de connaissance à propos de la position de la colonne considérée dans l'image courante.
2. Le module de stratégie globale collecte les questions sur l'ensemble des pages du lot, et les soumet à une interface homme–machine adaptée permettant à un ou plusieurs opérateurs humains de valider ou corriger la position de la colonne contenant les numéros de ventes dans les images problématiques.
3. Le module de stratégie globale collecte les réponses aux questions, et invoque à nouveau le module d'interprétation de page avec ces nouvelles informations afin de permettre la production de nouveaux résultats basés sur une localisation fiable de la colonne.

Dans ce scénario, quelques actions d'un opérateur humain permettent une remise en cause en profondeur des résultats produits par le module d'interprétation de page. L'information apportée au module d'interprétation de page par son environnement permet de compléter efficacement sa connaissance en corrigeant la cause du problème constaté, et non ses multiples conséquences (position et valeur de chaque numéro de vente). Finalement, on peut noter que ce scénario peut être fusionné avec le précédent pour proposer une approche

plus réaliste de l'interprétation de ces tableaux de ventes, réclamant une confirmation pour les éléments les plus structurants, et permettant un ajustement des éléments terminaux ensuite.

Cette section a montré que la mise en place d'un interprétation contextuelle et assistée d'un fonds documentaire pouvait se baser sur le même mécanisme itératif, et que les modes d'interaction était compatibles. Ceci permet l'intégration homogène d'informations externes à la page lors de l'interprétation de cette dernière, que cette information soit extraite d'autres pages ou fournie par un opérateur humain. Par ailleurs, nous avons également vu que la séparation des connaissances exprimées par le concepteur au niveau du module d'interprétation de page et du module de stratégie globale permet d'appréhender plus simplement la mise en place de chacun de ces modules.

5.3 Conclusion

Ce chapitre vient de présenter une vue d'ensemble du système d'interprétation de fonds documentaires que nous proposons. Le mécanisme d'interprétation itératif global, qui permet l'intégration homogène d'informations contextuelles lors du traitement d'une page, est basé sur :

1. une architecture à deux niveaux d'abstraction, où le niveau global, représenté par le module de stratégie globale, permet la mise en place de circulations d'information complexes, et où la base de données centrale permet l'enrichissement progressive de la connaissance extraite du fonds, tout en autorisant des échanges asynchrones entre les modules ;
2. la capacité du module d'interprétation de page à accepter et exploiter des informations produites par son environnement, ce qui est géré par la mémoire visuelle et la description de la page, et à réclamer, lorsque c'est possible, la connaissance qui lui manque (nous reviendrons en détail sur ces points dans le prochain chapitre).

Ceci n'a d'intérêt que parce que la conception d'un système spécifique reste simple, en particulier grâce à la séparation claire entre :

- le niveau de la page, chargé d'extraire l'information contenue dans l'image en tirant profit de connaissances externes, sans se préoccuper de comment ces informations sont générées ni d'où elles proviennent ;
- le niveau global, chargé de gérer la circulation de l'information et l'exploitation de propriétés globales, sans se préoccuper de comment l'information est effectivement extraite.

L'exploitation simultanée des propriétés du fonds documentaire et des informations apportées par les opérateurs humains a deux avantages majeurs. D'une part, elle autorise une certaine efficacité dans la sollicitation des opérateurs humains, car il est possible de trier les questions, les filtrer ou les regrouper, selon la vision globale de la connaissance du fonds (dans la base de données centrale) à un instant donné. D'autre part, il est possible de mettre en place simplement (car il suffit de combiner des modules élémentaires) des scénarios complexes. On peut, par exemple, penser à :

- la validation ou la correction d'éléments incertains (ex : nombres peu fiables détectés dans plusieurs pages et transmis à un opérateur humain) ;
- le guidage de l'interprétation (ex : correction de la position d'une colonne dans un tableau, qui remet en cause toute l'extraction d'information liée) ;

- l'optimisation de résultats locaux (hypothèses de reconnaissance) grâce à l'exploitation de contraintes sémantiques (ex : séquences de numéros de ventes qui respectent un incrément d'une unité entre chaque occurrence) ;
- l'accélération de saisies manuelles grâce au regroupement d'éléments visuellement similaires (ex : en faisant des clusters de mots qui se ressemblent sur plusieurs pages voisines, et en présentant ces clusters à des opérateurs humains) ;
- l'enrichissement automatique d'un lexique lorsque des nouveaux mots sont reconnus et validés (ce qui permet un augmentation rapide des taux de reconnaissance) ;
- l'ajustement progressif de paramètres des modèles de pages (taille des colonnes dans un tableau, par exemple) en exploitant la structure des pages bien reconnues pour guider l'interprétation de celles plus difficiles ;
- l'apprentissage de modèles structurés de page à partir d'exemples de décompositions de pages construits semi-automatiquement...

Ces deux derniers éléments dépassent le cadre de nos travaux, mais font partie des objectifs vers lesquels notre approche tend.

Un autre élément intéressant de cette architecture est sa capacité à gérer automatiquement, au niveau du module de stratégie globale, la quantité de travail demandée aux opérateurs humains, et donc le rapport entre la qualité des données produites, et leur coût, à condition de disposer d'un dispositif d'évaluation adapté.

On peut également remarquer que, dans le cas de fonds documentaires contenant peu d'éléments, notre approche semble également adaptée car elle permet de répartir simplement l'effort humain entre la phase de conception et la phase d'exécution, selon que le concepteur prenne le temps d'anticiper des cas particuliers, ou qu'il favorise une majorité de traitements manuels.

Pour compléter la présentation de nos propositions, nous allons détailler dans le prochain chapitre comment étendre un module d'interprétation de page existant pour systématiser : (i) la fusion de données externes avec les données extraites de l'image (mécanisme de mémoire visuelle) ; (ii) la gestion de l'échange d'informations avec l'environnement du module, grâce à un extension ciblée du langage de description d'une page qui permet d'exprimer de nouvelles propriétés utiles.

Chapitre 6

Conception d'un module d'interprétation de page interactif

Introduction

Dans ce chapitre, nous focalisons notre présentation sur les transformations à apporter à un module d'interprétation de page pour lui permettre de tirer profit d'information contextuelle produite par son *environnement*, c'est à dire extraite du fonds documentaire ou fournies par un opérateur humain. Nous montrons qu'il est possible de systématiser l'échange d'information entre le niveau de la page et le niveau global en enrichissant la description du contenu de l'image à analyser. Autrement dit, grâce à l'expression déclarative de nouvelles propriétés à propos de la page, nous proposons de mettre en place automatiquement une interaction entre le module d'interprétation de page et le module de stratégie globale (présentés dans le chapitre précédent) qui permet une détection et une correction d'erreur asynchrone au cours des itérations du processus global d'interprétation du fonds documentaire.

Avant de débuter ce chapitre, nous attirons l'attention du lecteur sur trois points importants.

1. Nous rappelons que nous nous intéressons à décrire les modifications à apporter à un système existant, et non à proposer un nouveau système complet. Nous ferons des propositions d'extensions sur la base d'un module d'interprétation de page élémentaire, respectant un travail image par image que nous ne souhaitons pas perturber ni dans la conception, ni dans le fonctionnement.
2. Ce chapitre contient un certain nombre d'éléments formels, dont on peut questionner l'intérêt à première vue, car nos propositions ont été validées par la réalisation d'un système concret, et son évaluation avec des scénarios spécifiques. Notre motivation est double ici : tout d'abord nous avons souhaité fournir, en plus des explications textuelles *qui suffisent à la compréhension de nos propositions*, une description précise et non ambiguë de nos travaux ; et, par ailleurs, nous avons souhaité faciliter la réutilisation de nos travaux en les exprimant d'une manière indépendante d'un quelconque système.
3. Finalement, afin de nous concentrer sur l'illustration des points essentiels, les exemples de ce chapitre ne tirent pas pleinement profit de l'asynchronisme possible dans les échanges entre les parties du système, et s'intéressent à l'évolution de la mémoire visuelle lors des itérations successives pour *une* page. Il faudra toutefois garder à l'es-

prit que ce qui est ici présenté pour une page s'applique simultanément à un grand nombre d'entre elles lors de l'exploitation du système.

Ces précautions étant prises, voyons à présent à la structure de ce chapitre.

- La première section s'intéresse à la mise en place d'un canal de communication entre le module d'interprétation de page et son environnement, ainsi qu'à la fusion de l'information contextuelle avec celle extraite de l'image grâce à la structure de mémoire visuelle. Cette section formule des propositions à un niveau algorithmique.
- La deuxième section est un détour obligé par la formalisation d'un module d'interprétation de page basique qui constitue une base théorique pour la génération automatique d'un programme d'interprétation à partir de la description d'une page, et l'intégration de la gestion de la mémoire visuelle.
- La troisième section pousse, quant à elle, la présentation un cran plus loin, en décrivant comment intégrer de nouveaux opérateurs qui viennent enrichir notre vocabulaire pour décrire le contenu de la page. La compilation de ces nouveaux opérateurs permet une gestion automatique, en cours d'interprétation, des erreurs grâce à la sollicitation ou à l'exploitation d'informations contextuelles provenant de l'environnement du module, au travers d'une communication asynchrone.

6.1 Communiquer avec l'environnement pour activer l'interaction

Dans cette section, nous présentons comment rendre un module d'interprétation de page capable :

- d'échanger des informations de façon asynchrone avec son environnement ;
- d'utiliser ces dernières au cours de son travail d'interprétation d'une image, pour éventuellement modifier son comportement automatique.

Nous présentons d'abord le modèle interactif sur lequel nous nous sommes basés, puis nous montrons comment la structure de mémoire visuelle permet d'être le pivot dans la réintégration d'informations contextuelles externes lors de l'interprétation d'une image.

6.1.1 Modèle théorique de l'interaction

La base du mécanisme itératif d'interprétation mis en place au niveau du module d'interprétation de page est largement inspiré du modèle de *machine de Turing persistante* formalisé par Goldin et Wegner [124, 41, 125]. Ces machines permettent, selon ces auteurs, d'étendre le modèle de *machine de Turing* pour rendre possible la représentation de systèmes interactifs. Nous présentons ici rapidement ce modèle, et l'application que nous en faisons.

6.1.1.1 Machines de Turing persistantes

Goldin et Wegner proposent la définition suivante des *machines de Turing*.

> Les *machines de Turing* sont un dispositif fini de calcul qui transforme une chaîne d'entrée en une chaîne de sortie par une séquence de transitions entre des états.

Les auteurs indiquent que le modèle communément utilisé est celui d'une machine possédant un ruban en lecture seule, dit le *ruban d'entrée*, un ruban en écriture seule, dit le

ruban de sortie et un ou plusieurs *rubans de travail*. La propriété de ce type de dispositif est de ne considérer son environnement que lors de l'initialisation, c'est à dire en lisant l'entrée, et ensuite la machine qui démarre toujours dans le même état produit un résultat indépendamment de son environnement.

Afin de pouvoir tirer profit d'informations contenues dans l'environnement de la machine, et réagir à cette dernière, les auteurs indiquent qu'il est nécessaire de rendre le dispositif *interactif*, et donnent la définition suivante.

> Un *dispositif de calcul interactif* autorise des actions d'entrée et de sortie pendant le processus de calcul.

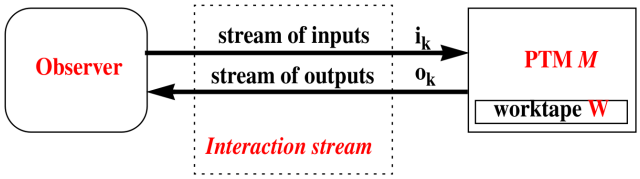


FIGURE 6.1 – Schéma de *machine de Turing persistante*, d'après [41].

L'extension proposée par Goldin et Wegner, illustrée en figure 6.1, est alors définie comme suit.

> Les *machines de Turing persistantes* sont des machines de Turing à plusieurs rubans, avec un ruban persistant entre les étapes du calcul de la machine. Ce ruban spécial est appelé *mémoire*, et le contenu de cette mémoire avant et après une étape de calcul est appelé *état*.

L'*état* de cette nouvelle machine n'appartient pas à un espace fini et connu comme dans le cas des machines de Turing classiques : il est construit à partir d'informations externes, et peut dépendre de leur ordre d'arrivée. L'état de départ lors d'un calcul n'est pas toujours le même, et les auteurs indiquent qu'une *machine de Turing interactive M* définit une *fonction partielle récursive* f_M telle que :

$$f_M : (\text{Entrée, Mémoire}) \rightarrow (\text{Sortie, Mémoire}) \tag{6.1}$$

Les auteurs donnent l'exemple d'un système de répondeur R. Les opérations possibles sont *enregistrer(message)*, *jouer* et *supprimer*, et la mémoire est une file de messages F. Le système est alors défini par [1] :

$$\begin{aligned} f_R(\textit{enregistrer}(e),\ F) &= (\texttt{ok},\ \textit{ajouter\_fin}(e,\ F))\ ; \\ f_R(\textit{jouer},\ F) &= (\textit{tete}(F),\ F)\ ; \\ f_R(\textit{supprimer},\ F) &= (\texttt{ok},\ \textit{supprimer\_debut}(F)). \end{aligned} \tag{6.2}$$

6.1.1.2 Contrainte d'asynchronisme : vers une pseudo interaction

Ce modèle nous est apparu comme particulièrement intéressant car il correspond bien au besoin de réintégration d'information externe lors de l'interprétation itérative d'une image. Il fournit une base théorique qui autorise à raisonner à propos du comportement de la machine, en particulier en analysant les *traces* formées par la succession d'états observables de la mémoire au cours des étapes d'un calcul.

1. Nous avons simplifié l'exemple, qui ne gère pas ici le cas particulier d'une file vide.

Toutefois, notre cadre applicatif a limité l'application directe de ce modèle. En effet, l'impossibilité de communiquer directement avec chaque module d'interprétation de page en cours d'exécution pour chaque image traitée dans le système impose de réfléchir à une adaptation qui :

1. soit dédiée à l'interprétation d'images ;
2. autorise l'apport simultané de plusieurs informations à propos d'une image ;
3. évite la dépendance à l'ordre d'apport des informations au cours des différentes étapes du calcul (ce type de système pourrait y être sensible, or les résultats du module d'interprétation de page ne doivent pas dépendre de l'ordre dans lequel un opérateur humain a corrigé les éventuels problèmes) ;
4. permette une sauvegarde et un chargement de la mémoire associée à l'interprétation d'une image donnée lors de la terminaison ou de la reprise d'un module d'interprétation de page ;
5. exploite simultanément les informations provenant de l'image et celles provenant de l'environnement pour proposer une nouvelle interprétation.

6.1.2 Fusion d'informations externes grâce à la mémoire visuelle

Pour répondre aux besoins pratiques formulés dans la sous-section précédente, nous avons mis en place un protocole simplifié d'interaction où les étapes de calcul sont longues et correspondent à l'interprétation (partielle ou complète) de toute une image. Nous avons utilisé la structure de *mémoire visuelle* définie en sous-section 5.1.2.1. Cette structure, nous le rappelons, permet de stocker des informations de types variés à propos du contenu d'une image, et d'y accéder à partir de leurs caractéristiques géométriques (forme et position dans l'image) et de leur type. Comme présenté en sous-section 5.1.2.1, la mémoire visuelle associée à une image est donc une structure associative contenant des triplets du type suivant :

$$(\text{Zone} \times \text{TypeDonnée}) \rightarrow \text{Informations} \tag{6.3}$$

Nous nous intéressons ici à l'adaptation que nous avons faite du modèle de *machine de Turing interactive* pour permettre la mise en place d'une communication entre le module d'interprétation de page et son environnement, et comment ce dernier peut exploiter l'information contextuelle reçue.

6.1.2.1 Adaptation du modèle de machine de Turing interactive

Dans l'hypothèse d'une application stricte à notre problème du modèle de machine de Turing interactive présenté précédemment, l'utilisation de la structure de *mémoire visuelle* comme support de la mémoire de la machine entraîne la mise en place des opérations suivantes au niveau du module d'interprétation de page :

- *interpréter(image)*, pour lancer l'interprétation d'une image ;
- *ajouter_contexte(zone, type, info)*, pour ajouter un ensemble d'information contextuelles dans la mémoire ;
- *supprimer_contexte(zone, type, info)*, pour supprimer certains éléments de la mémoire.

La gestion de la mémoire du module (noté *MIP*) serait alors décrite par des fonctions semblables à celles qui suivent (on note D un triplet *(zone, type, info)*) :

$$\begin{array}{ll} f_{MIP}(\textit{interpréter}(I),\ M_I(t)) & = (\text{résultat},\ M_I(t+1))\ ; \\ f_{MIP}(\textit{ajouter\_contexte}(D),\ M_I(t)) & = (\texttt{ok},\ M_I(t) \cup D)\ ; \\ f_{MIP}(\textit{supprimer\_contexte}(D),\ M_I(t)) & = (\texttt{ok},\ M_I(t) \setminus D). \end{array} \tag{6.4}$$

En pratique, le module d'interprétation de page n'est pas un processus persistant, et le stockage de la mémoire visuelle se fait au niveau de la base de données centrale. Par conséquent, il est plus pratique de laisser au module de stratégie globale la responsabilité de l'ajout et suppression d'éléments en base, car il est le garant de la cohérence de l'interprétation globale. Le module d'interprétation de page peut transformer le contenu de la *mémoire visuelle* associée à une image donnée, mais le module de stratégie globale peut également modifier les informations contenues et choisir quelles informations y stocker pour améliorer le comportement du module d'interprétation de page pour l'image concernée.

De cette façon, la définition de l'équation 6.4 est réduite pour donner en pratique :

$$f_{MIP}(\textit{interpréter}(I),\ M_I(t)) = (\text{résultat},\ \Delta M_I(t,\ t+1)). \tag{6.5}$$

Dans la mesure où le choix de conserver ou non les modifications faites par le module d'interprétation de page est de la responsabilité du module de stratégie globale, le module d'interprétation de page retourne une liste des modifications (notée $\Delta M_I(t,\ t+1)$) qu'il propose pour le contenu de la mémoire visuelle entre l'instant t et l'instant $t+1$. La fusion de ces modification avec la version stockée dans la base de données centrale est alors gérée par le module de stratégie globale.

6.1.2.2 Définition d'un protocole d'interaction asynchrone

Notre protocole d'échange d'informations est donc le suivant :

1. le module de stratégie globale identifie une image de page (I) à interpréter ou réinterpréter, et extrait l'information utile ($M_I(t)$) de la base de donnée centrale ;
2. il invoque le module d'interprétation de page en lui passant l'image en question et l'ensemble des informations (contextuelles ou précédemment extraites et conservées) sous forme d'une *mémoire visuelle* ;
3. le module d'interprétation de page exploite les informations qu'il extrait de l'image et celles présentes dans la mémoire visuelle pour proposer une interprétation, partielle ou complète, selon ce qu'il est capable de faire, et un ensemble de modifications à apporter à la mémoire visuelle (couple $(\text{résultat}, \Delta M_I(t,t+1))$) ;
4. le module de stratégie globale exploite ces nouvelles données selon le scénario défini pour traiter ce fonds documentaire, en général en les stockant dans la base centrale et en déclenchant d'autres traitements si nécessaire, de la même manière.

Ce protocole fait les hypothèses suivantes à propos du module d'interprétation de page :

1. il est en mesure d'exploiter *directement* l'information contenue dans la mémoire visuelle : c'est le format pivot permettant la fusion d'informations contextuelles et d'informations extraites de l'image ;
2. il redémarre *complètement* son interprétation pour prendre en compte ces nouvelles informations et proposer une réponse plus adaptée.

L'item 1 impose de s'intéresser aux structures de données internes du module d'interprétation de page, que nous présentons dans la sous-section suivante. L'item 2 impose, quant à lui, de savoir quelles données enregistrer dans cette mémoire visuelle et comment y réagir. Il ne nous semble pas pertinent de stocker tous les résultats intermédiaires générés par le module d'interprétation de page, ou le résultat le plus abouti auquel ce dernier est arrivé, car nous souhaitons pouvoir remettre en cause l'interprétation qu'il propose, et par conséquent ses résultats intermédiaires. Nous avons identifié deux catégories d'informations dont l'enregistrement dans cette mémoire visuelle peut être utile :

1. celles dont la production nécessite un calcul long avec peu de contexte, comme des mots reconnus par OCR : nous verrons dans la sous-section 6.1.2.4 comment les conserver ;
2. celles qui permettent de modifier le comportement du mécanisme d'interprétation automatique, comme la position d'éléments structurants qui ne seraient pas détectés (filet, virgule, etc.) : nous verrons dans la section 6.2 comment réclamer ou utiliser automatiquement ces éléments.

6.1.2.3 Exploitation des données externes

Nous décrivons ici l'organisation interne du module d'interprétation de page, en ce qui concerne sa capacité à utiliser les informations contenues dans la mémoire visuelle qui lui est fournie lors de son invocation.

Mise en œuvre de la mémoire visuelle Cette mémoire visuelle doit, à l'instar de l'information extraite du signal de l'image, être accessible à toute fonction du programme d'interprétation. Nous proposons donc d'utiliser la notion de *calque* proposée par Lemaitre [61] (deux calques sont nécessaires pour mettre en place la mémoire visuelle, mais nous y reviendrons). Lemaitre s'est servi de cette structure pour mettre en place un mécanisme d'interprétation perceptive fusionnant l'information extraite à différents niveaux de résolution d'une image, stockant alors des primitives visuelles dans ces calques : traits, segments ou composantes connexes. Dans notre cas, nous étendons ce modèle pour stocker tout type d'information pour laquelle on fournit des caractéristiques géométriques et un type. De cette manière, nous disposons d'un ensemble de structures capables de stocker des informations possédant des caractéristiques géométriques dans le référentiel de l'image courante.

La mémoire visuelle mise en œuvre dans le module d'interprétation de page est formée des deux calques suivants :

1. le *calque de travail*, noté M_T, initialisé avec la mémoire visuelle fournie à l'invocation du module ;
2. le *calque des changements*, noté M_C, qui collecte les modifications que le module fait sur cette structure, et sert à renvoyer $\Delta M_I(t, t+1)$ au module appelant.

Le module d'interprétation de page est alors capable d'exploiter ces informations en *sélectionnant le calque approprié*, en se *focalisant sur une zone de recherche*, et en *manipulant les données stockées*.

Types de données Nous utiliserons ces deux types de données définis dans la suite :

$$\begin{array}{lcl} \text{DonnéeImage} & \equiv & (\text{Zone} \times \text{TypeDonnée}) \\ \text{DonnéeMémoire} & \equiv & (\text{Zone} \times \text{TypeDonnée}) \rightarrow \text{Informations}) \end{array} \tag{6.6}$$

Les type de référence sont définis comme suit :

Zone domaine des surfaces incluses dans le référentiel de l'image.
Un élément de ce type permet de décrire les caractéristiques géométrique d'un objet, par exemple avec son rectangle englobant.

TypeDonnée domaine des marqueurs (ou types) définis par le concepteur du système, qui permettent de décrire la nature des objets stockés en mémoire (et en interpréter le contenu).
Quelques exemples de ces types d'objets « mémorisables » peuvent être : `nombre`, `patronyme`, ou encore `colonne_numero_ordre`.

Informations domaine des tables associatives.
En pratique, il s'agit d'un ensemble de couples $(attribut \rightarrow valeur)$ qui permet de stocker le contenu d'un objet structuré du type décrit dans le paramètre précédent : hypothèses de reconnaissance, valeur finale, etc.

Opérations sur la mémoire visuelle Quatre opérations primitives sont possibles sur les données mémorisées, et ont les conséquences suivantes sur le contenu des calques.

lire_mem(zone, type)
Type : $(\text{Zone} \times \text{TypeDonnée}) \rightarrow \text{DonnéeMémoire}$
Cette opération consulte le contenu du calque de travail dans la zone indiquée, et si une donnée du bon type y est présente, renvoie cette donnée.

consommer_mem(zone, type)
Type : $(\text{Zone} \times \text{TypeDonnée}) \rightarrow \text{DonnéeMémoire}$
Cette opération consulte le contenu du calque de travail dans la zone indiquée, et si une donnée du bon type y est présente, elle la supprime du calque de travail et renvoie cette donnée.

ajouter_mem(zone, type, infos)
Type : $(\text{Zone} \times \text{TypeDonnée} \times \text{Informations}) \rightarrow \text{DonnéeMémoire}$
Cette opération ajoute une donnée dans le calque de changements en lui associant les caractéristiques géométriques, le type et les informations indiquées [2]. Pour faciliter l'écriture de programmes, la fonction retourne la donnée qui vient d'être créée.

supprimer_mem(zone, type, infos)
Type : $(\text{Zone} \times \text{TypeDonnée} \times \text{Informations}) \rightarrow \text{DonnéeMémoire}$
Cette opération supprime une donnée du calque de travail, ajoute dans le calque de changements un marqueur spécial (`asupprimer(donnée)`) indiquant que la donnée a été marquée pour suppression [3], et renvoie la donnée supprimée.

Pour chacune des fonctions précédentes, il est donc nécessaire d'ajouter en paramètre des références vers les calques servant à stocker l'information mémorisée. Dans la suite, nous considérerons qu'on peut abstraire cette structure à deux calques en ajoutant simplement un paramètre $M = (M_T, M_C)$ représentant la structure mémoire.

2. Il serait possible d'ajouter également cette donnée dans le calque de travail, si on souhaite qu'elle puisse être exploitée lors de l'itération courante. Ceci requiert alors un paramètre supplémentaire.

3. En pratique, l'utilisation d'identifiants uniques pour les données mémorisées simplifie le travail, mais la gestion de méta-données alourdirait la notation sans intérêt pour la compréhension.

6.1.2.4 Mémoriser des résultats

Comme nous l'avons indiqué précédemment, chaque invocation du module d'interprétation de page reprend le travail d'interprétation de l'image dès le début, et ce dernier se base sur le contenu de la mémoire visuelle pour adapter sa réponse, et éventuellement éviter de refaire des calculs inutiles. Grâce aux outils primitifs que nous venons de présenter, il est déjà possible de modifier le comportement de ce module pour lui éviter, par exemple, de relancer un classifieur sur une zone donnée, et d'utiliser à la place la valeur enregistrée en mémoire. Pour mettre en place ce type de comportement, on peut recourir à un modèle de *raisonnement par défaut systématique*, qui vérifie qu'une information satisfaisante est présente en mémoire avant d'invoquer une fonction automatique. Supposons qu'un programme de lecture d'adresse postale contienne une définition de la forme suivante :

$$\mathit{fin\_adresse}(\mathit{zoneLigne}) = \mathit{localiser\_reconnaitre\_code\_postal}(\mathit{zoneLigne})$$

où *localiser_reconnaitre_code_postal* retourne un couple (*positionCode*, *valeurCode*). Dans ce cas, il est possible, en utilisant la mémoire visuelle, de transformer l'appel précédent par :

$$\begin{aligned}\mathit{fin\_adresse}(\mathit{zoneLigne}) = \mathit{conserver\_resultat}(&\mathit{zoneLigne},\\ &\texttt{code\_postal},\\ &\mathit{localiser\_reconnaitre\_code\_postal})\end{aligned}$$

L'élément `code_postal` doit être ajouté par le concepteur à la liste des éléments qui définissent le domaine TypeDonnée, et servira à indiquer dans la mémoire visuelle le type de donnée contenu.

La fonction *conserver_resultat* peut simplement être mise en place :

- par une construction d'ordre supérieur dans un langage fonctionnel ou logique ;
- par un mécanisme d'annotation dans un langage impératif : « annotations » Java, « décorateurs » Python, etc.

Le type de cette fonction est alors :

$$\underset{\mathit{zoneRecherche}}{\text{Zone}} \times \underset{\mathit{type}}{\text{TypeDonnée}} \times (\text{Zone} \to (\text{Zone} \times \text{A})) \to (\text{Zone} \times \text{A}) \qquad (6.7)$$

conserver_resultat prend en paramètre :

1. la zone à transmettre à la fonction f_{auto}, qui sert à rechercher une donnée satisfaisante dans la mémoire visuelle ;
2. un élément de type TypeDonnée qui permettra d'identifier le type de contenu à stocker dans la mémoire visuelle ;
3. une fonction f_{auto}, qui à une zone de recherche (de type Zone) renvoie une localisation précise (également de type Zone) et une valeur (d'un type A quelconque) ;

et elle renvoie le résultat de l'appel à la fonction f_{auto}.

Une définition possible de cette fonction *conserver_resultat* est donnée en algorithme 1, et peut être exprimée de la façon suivante : *« si un résultat acceptable existe en mémoire, alors le programme l'utilise, sinon il calcule le résultat avec la méthode automatique et conserve ce dernier avant de l'utiliser. »*

L'algorithme que nous venons de présenter est une variante de la technique dite de « mémoization » [78] où les résultats sont normalement enregistrés dans une structure liée à

Algorithm 1 Définition possible de la fonction *conserver_resultat*.

function CONSERVER_RESULTAT(*zoneRecherche*, *type*, f_{auto})
  $m \leftarrow lire\_mem(zoneRecherche, type)$
  if $m \neq$ **null then**
    return $(m.zone, m.info)$
  else
    $(zoneResultat, resultat) \leftarrow f_{\text{auto}}(zoneRecherche)$
    $ajouter\_mem(zoneResultat, type, resultat)$
    return $(zoneResultat, resultat)$
  end if
end function

la définition de la fonction. Ici, les résultats ne sont pas associés à une fonction précise, mais à une position dans l'image [4] pour pouvoir partager cette information avec l'environnement, si nécessaire.

C'est en autorisant la modification d'informations de ce type à l'extérieur du module d'interprétation de page que nous permettons de :

- corriger certaines erreurs, comme la valeur d'un nombre par exemple, grâce à l'exploitation d'un contexte plus large, et de réintégrer ces informations automatiquement dans la structure résultat, aussi complexe soit-elle ;
- interférer de façon assez naturelle avec le comportement par défaut du module, en lui apportant de nouvelles informations *à propos du contenu de l'image*, qui seront utilisées au lieu d'utiliser l'algorithme automatique d'interprétation d'image.

Nous développerons ces points dans la section suivante.

6.1.2.5 Vers l'exploitation systématique de propriétés des contenus

L'exemple précédent permet également d'introduire deux autres aspects de notre contribution.

Au niveau de la description du contenu d'une page, tout d'abord, on peut remarquer que cette première fonction, visant à modifier le comportement du module d'interprétation de page sans perturber le sens des résultats, peut être automatiquement utilisée dès que des calculs coûteux en temps sont réalisés pour l'interprétation avec peu de contexte de certains éléments. Dans le cas reconnaissance de nombre, par exemple, on se contente généralement de transmettre aux classifieurs seulement l'information relative au fragment de signal concerné. Par conséquent, une description déclarative du contenu de l'image peut indiquer, pour ce type de cas, qu'une partie de la description est *hors contexte, coûteuse en temps de calcul*. Il sera alors possible de gérer automatiquement cette information en générant un appel systématique à la fonction *conserver_resultat*. C'est donc un premier exemple d'exploitation automatique de certaines propriétés du contenu.

L'autre aspect sur lequel nous souhaitons insister est le manque de clarté dans l'évolution de l'état de la mémoire visuelle dans l'exemple précédent, et la nécessité d'une formalisation plus précise pour permettre l'adaptation d'un système existant, dès lors qu'une description complète génère une composition d'un grand nombre de fonctions d'analyse.

4. Toutefois, rien n'empêcherait de les rendre plus spécifiques, et le type des données peut déjà être très discriminants.

Il sera nécessaire de s'attarder sur cette formalisation dans la section suivante qui montrera comment modifier le flot de contrôle du module d'interprétation de page, et quelles seront les structures de données nécessaires.

Dans cette section, nous avons montré que grâce à l'adaptation du modèle de machine de Turing persistante que nous proposons et à la structure de mémoire visuelle, il est possible de tirer profit d'informations contextuelles, générées à l'extérieur du module d'interprétation de page, lors de l'exécution de ce dernier. Il est également possible de modifier le comportement automatique de ce module si on modifie le contenu de cette mémoire visuelle entre les invocations du module d'interprétation de page par le module de stratégie globale.

Notons enfin que les approches existantes basées sur une expression algorithmique de la connaissance à priori à propos des pages pourront tirer profit de ces premières propositions pour lesquelles aucune forme de description du contenu des pages n'est nécessaire. Cependant, comme nous allons le montrer dans les sections suivantes, il est également possible de proposer des extensions du langage de description, pour les approches concernées, afin de systématiser la production et l'utilisation d'informations pour interagir avec l'environnement, et laisser à ce dernier la possibilité de modifier les données de la mémoire pour interférer avec le comportement automatique du module d'interprétation de page afin d'améliorer ses réponses.

6.2 Formalisation d'un module de référence

Cette seconde section détaille comment systématiser la mise en place d'un échange d'informations asynchrone entre le module d'interprétation de page et son environnement pour permettre de réintégrer et exploiter des informations contextuelles pendant l'interprétation de chaque page. Au delà de l'objectif de diminution de l'effort nécessaire à la production de résultats corrects, cette gestion automatique des échanges permet de limiter l'effort de conception du système, car le concepteur va simplement exprimer des propriétés à propos du contenu des pages, sans se préoccuper de la façon dont les échanges seront effectivement mis en place.

L'idée fondamentale est alors de permettre au concepteur du système d'utiliser des outils génériques pour décrire le contenu de la page de façon « synchrone », et de générer automatiquement, grâce à une compilation de la description fournie, le programme au cœur du module d'interprétation de page, spécifique à un type de page donné, gérant une communication asynchrone, comme le montre le schéma de la figure 6.2.

Nous nous intéressons donc ici à des outils qui permettent de :

1. décrire le contenu des documents grâce à un langage de description défini par sa *syntaxe* ;
2. générer automatiquement un programme exécutable en donnant une *sémantique* (au sens de la théorie des langages de programmation) à la description du document.

Dans la mesure où notre travail ne consiste pas à proposer intégralement un nouveau système d'interprétation de documents, mais plutôt de proposer des extensions à apporter à des systèmes existants pour leur permettre de tirer facilement profit d'informations contextuelles pendant la phase d'interprétation, nous avons besoin de définir clairement les prérequis des approches qui pourront être adaptées, avant de spécifier précisément les modifications que nous proposons. Pour répondre à ce double objectif, nous avons eu besoin d'un

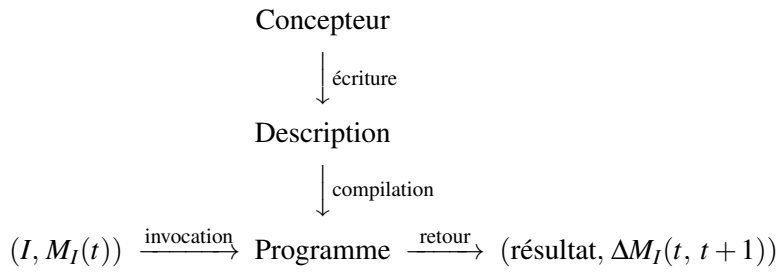


FIGURE 6.2 – Production d'un programme d'interprétation de page à partir d'une description.

formalisme à la fois flexible et efficace pour exprimer le fonctionnement du programme généré après compilation d'une description.

Dans cette sous-section, nous commençons par présenter le formalisme nécessaire pour décrire la sémantique des programmes produits, puis nous donnons une description formelle du module de référence que nous étendrons et des opérations sur la mémoire visuelle nécessaires.

Ces modifications permettront de mettre en place, dans la section suivante les différents modes d'interaction (dirigée et spontanée) présentés dans le chapitre précédent, tout en construisant progressivement des niveaux d'abstraction destinés à faciliter l'écriture de descriptions par le concepteur, ou à garantir dans une certaine mesure la progression de l'interprétation.

6.2.1 Notion de continuation et notations utilisées

Pour exprimer le *sens* des descriptions des pages, nous avons recours à une formalisation des états du programme d'interprétation, et pour ce faire nous allons exprimer (partiellement) sa *sémantique dénotationnelle*[5] à l'aide d'une représentation par *passage de continuation*.

Selon Ridoux [102], une *continuation* peut être définie comme *« [une] structure de données abstraite qui permet de formaliser le contrôle des langages de programmation en notant les calculs qui restent à faire. »* Dans le cas d'une sémantique dénotationnelle, dite par passage de continuation, le sens mathématique d'un programme est une fonction qui prend une continuation (celle qui suit son exécution) et rend une continuation (celle qui correspond à son exécution). Cette approche est particulièrement intéressante dans notre travail car elle permet :

- de composer simplement des opérations pour définir le comportement du programme, et de définir de façon macroscopique les éléments requis, ce qui va dans le sens de notre démarche d'extension d'un système existant ;
- d'implémenter rapidement un compilateur qui transforme le langage de description en un programme d'interprétation de page, en spécifiant précisément l'état du programme à chaque étape de son calcul, ainsi que son flot de contrôle ;

5. Introduite par Scott et Strachey [109], cette formalisation du *sens* d'un programme est basée sur la représentation de ce dernier par une composition de fonctions mathématiques. Elle nous semble adaptée à la construction d'un compilateur comme le montrent les travaux récents sur les combinateurs d'analyseurs syntaxiques. Voir [40] pour une introduction rapide.

– d'utiliser explicitement la notion de *calcul futur* bien adaptée au problème d'écriture de programmes intégrant des interactions asynchrones.

Nous allons donner deux exemples pour préciser ces notions et ce formalisme, avant de proposer une formalisation du programme de base du module d'interprétation de page que nous allons étendre dans la suite de cette section.

6.2.1.1 Premier exemple : notion de continuation

Commençons par préciser la notion de continuation : l'équation suivante montre la définition d'une fonction simple :

$$\begin{array}{rccc} \mathit{incrementer} : & \mathbb{N} & \rightarrow & \mathbb{N} \\ & x & \mapsto & x+1 \end{array} \tag{6.8}$$

Une invocation de cette fonction [6] peut alors être :

$$(\mathit{incrementer}\ \ 3) \tag{6.9}$$

Une façon de transformer cette définition pour lui permettre d'accepter une *continuation* κ est la suivante :

$$\begin{array}{rccccc} \mathit{incrementer} : & \mathbb{N} & \times & (\mathbb{N} \rightarrow \mathrm{Cont}) & \rightarrow & \mathrm{Cont} \\ & x & , & \kappa & \mapsto & (\kappa\ (x+1)) \end{array} \tag{6.10}$$

Où on peut considérer que le type Cont est (État $\rightarrow$ État). Nous conserverons cette simplification dans la suite. L'invocation de cette fonction prend alors une forme telle que :

$$(\mathit{incrementer}\ \ 3\ \ \lambda x.(\mathit{print}(x))) \tag{6.11}$$

Où $\lambda x.(\mathit{print}(x))$ est une fonction qui accepte un paramètre et affiche la valeur de ce dernier sur la sortie standard. Elle permet de capturer l'état du programme à la fin du calcul.

Dans cette approche, une fonction ne retourne pas de résultat, mais elle applique la continuation sur son résultat. Les paramètres de la fonction de continuation (κ) passée à la fonction courante (*incrementer*) correspondent aux « résultats » produit par la fonction courante. Le contrôle du programme peut alors être réifié en devenant un ensemble de paramètres passés de fonction en fonction. Des techniques classiques existent pour éviter le débordement de la pile d'appel dans tous les langages de programmation fonctionnelle modernes.

6.2.1.2 Second exemple : compilation d'un analyseur syntaxique simple avec retour arrière

Pour illustrer l'intérêt de cette approche pour l'interprétation de documents, nous allons montrer comment construire un analyseur syntaxique simple permettant de reconnaître le langage $\mathcal{L} = \mathrm{a}^* \mathrm{b} \cup \mathrm{a}^* \mathrm{c}$. Nous allons nous baser sur une grammaire écrite de la façon suivante qui est assez naturelle, mais qui présente l'inconvénient de nécessiter un nombre illimité de

6. Nous utilisons ici une notation préfixe pour l'application d'une fonction. La syntaxe est la suivante : parenthèse ouvrante, référence à la fonction, liste de paramètres séparés par des espaces, parenthèse fermante.

lectures avant de prendre une décision lors d'une analyse descendante [7] :

$$\begin{array}{lcl} S & \longrightarrow & AB \mid AC \\ A & \longrightarrow & \mathbf{a}A \mid \mathbf{a} \\ B & \longrightarrow & \mathbf{b} \\ C & \longrightarrow & \mathbf{c} \end{array} \tag{6.12}$$

Ici, S représente l'axiome de la grammaire, les non-terminaux sont A, B et C, et les terminaux sont **a**, **b** et **c**. La syntaxe du langage de description utilisé ici est alors extrêmement simple et contient quatre constructions permettant de décrire les règles de production de la grammaire :

l'alternative notée $A \mid B$ indique que l'expression peut être formée de A ou de B ;

la concaténation notée AB indique que l'expression est formée de A suivi de B ;

la consommation d'un terminal notée **x**, indique la position d'un terminal « x » dans le mot ;

la dérivation notée $G \longrightarrow D$ indique une abstraction qui (dans le cas de langages hors contexte) indique que le non-terminal G résume l'expression D.

Ce type de grammaire est représentatif, selon nous, des difficultés présentes dans l'interprétation de contenus de documents complexes, comme les documents anciens ou dégradés, pour lesquels on doit considérer un préfixe d'analyse parfois important avant de pouvoir savoir ce qu'il représente. Un mécanisme de *retour arrière* est alors indispensable pour permettre à la fois l'écriture d'une description (grammaticale) intuitive, et permettre la détection de structures complexes. Il est donc nécessaire de permettre la génération automatique d'un analyseur à partir de la description du contenu de la page qui mette en œuvre ce comportement.

Grâce à un *système de réécriture,* il est possible de produire automatiquement une fonction réalisant le traitement souhaité. Le système de réécriture présenté ci-après (la fonction $\mathcal{T}$) est un exemple d'une telle transcription. Pour bien distinguer les éléments appartenant au domaine de la description qui reste à traduire, des éléments qui appartiennent au domaine des fonctions, il est d'usage d'isoler les fragments de description par des crochets doubles ($[\![$ et $]\!]$).

Pour mettre en place le *mécanisme de retour arrière,* nous utilisons ici deux continuations, à l'instar d'un solveur Prolog. La continuation κ est la *continuation de succès,* à laquelle est passé le reste de la structure à analyser lorsque l'analyse peut progresser, et la continuation ζ est la *continuation d'échec* invoquée lorsque l'analyse ne peut plus progresser pour l'hypothèse courante. F représente la séquence à analyser [8].

$$\begin{array}{lcl} \mathcal{T}[\![A \mid B]\!] & \equiv & \lambda \mathrm{F}\kappa\zeta.(\mathcal{T}[\![A]\!]\ \mathrm{F}\ \kappa\ (\mathcal{T}[\![B]\!]\ \mathrm{F}\ \kappa\ \zeta)) \\ \mathcal{T}[\![A\,B]\!] & \equiv & \lambda \mathrm{F}\kappa\zeta.(\mathcal{T}[\![A]\!]\ \mathrm{F}\ \lambda \mathrm{F}'.(\mathcal{T}[\![B]\!]\ \mathrm{F}'\ \kappa\ \zeta)\ \zeta) \\ \mathcal{T}[\![\mathbf{x}]\!] & \equiv & \lambda \mathrm{F}\kappa\zeta.\begin{cases} (\kappa\ \mathit{cdr}(\mathrm{F})) & \text{si } \mathit{car}(\mathrm{F}) = \mathbf{x} \\ \zeta & \text{sinon} \end{cases} \\ \mathcal{T}[\![G]\!] & \equiv & \mathcal{T}[\![D]\!] \qquad \text{si } G \longrightarrow D \end{array} \tag{6.13}$$

Grâce à ce système de réécriture, on peut générer un analyseur syntaxique pour la langage $\mathcal{L}$ précédent. Le table 6.1 détaille chacune des constructions de l'équation 6.13.

7. Ici, « descendante » fait référence à une analyse $LL(k)$, avec k fini mais non borné [3].

8. Cette séquence est munie de deux opérations élémentaires : *car* et *cdr* qui renvoient respectivement le premier élément de la séquence, et la séquence privée du premier élément.

TABLE 6.1 – Détail des constructions utilisées dans l'exemple d'analyseur syntaxique avec retour arrière de l'équation 6.13 (page 97).

Règle de réécriture	**Description de la sémantique choisie**
$\mathcal{T}[\![A \mid B]\!]$ *Alternative*	Pour reconnaître *A ou B*, le système de réécriture construit, à partir de la description une fonction qui prend trois paramètres : une séquence à reconnaître F, une continuation de succès κ et une continuation d'échec ζ. Cette fonction va transmettre F et κ à $\mathcal{T}[\![A]\!]$, en modifiant la continuation d'échec pour que $\mathcal{T}[\![B]\!]$ soit appelée si la reconnaissance de *A* échoue.
$\mathcal{T}[\![A\ B]\!]$ *Concaténation*	Pour reconnaître *A puis B*, le système de réécriture produit une fonction qui consistera à appeler $\mathcal{T}[\![A]\!]$ avec la séquence d'entrée F, et à poursuivre l'analyse des éléments restants F′ avec $\mathcal{T}[\![B]\!]$ si l'élément abstrait *A* peut être reconnu. Dans les deux cas, un échec de reconnaissance produira un appel de ζ qui reste inchangée.
$\mathcal{T}[\![\mathbf{x}]\!]$ *Terminal*	Pour valider la reconnaissance d'un élément terminal **x**, le système de réécriture produit une fonction qui vérifie que le prochain élément de la séquence F est bien celui attendu, et dans ce cas appelle la continuation de succès κ avec la séquence mise à jour (privée du premier élément). Dans le cas contraire, la continuation d'échec ζ est appelée. Contrairement aux opérations de contrôle précédentes qui construisent l'arbre de recherche de solution en générant des fonctions de continuation, cette opération appelle directement les continuations et réduit l'arbre de recherche.
$\mathcal{T}[\![G]\!]$ *Dérivation*	Cette règle de réécriture ne sert qu'à gérer les niveaux d'abstraction de la description, et à sauvegarder les traductions déjà réalisées.

Comment exploiter ces définitions en pratique ? Les fonctions produites par l'application du système de traduction décrit par la fonction $\mathcal{T}$ peuvent être directement implémentées avec un langage de programmation fonctionnelle, mais peuvent aussi être implémentée dans d'autres environnements, comme par exemple une machine à piles. Une invocation simple de la fonction $f_S = \mathcal{T}[\![S]\!]$ pour déterminer si un mot appartient ou non au langage $\mathcal{L}$ peut alors être :

$$\begin{array}{l}(f_S\ \texttt{"aaaaaaac"} \\ \quad \lambda f.(\mathit{print}\ \text{« Mot reconnu. »}) \\ \quad \lambda.(\mathit{print}\ \text{« Mot inconnu. »}))\end{array} \tag{6.14}$$

6.2.2 Formalisation des propriétés requises du module d'interprétation de page

Nous allons à présent utiliser le formalisme introduit précédemment pour formaliser les caractéristiques qu'un module d'interprétation de page doit posséder pour permettre l'intégration d'un mécanisme d'interprétation itératif à l'aide de la mémoire visuelle. L'identification de ces éléments est importante pour permettre de définir une classe de programmes qui pourront être facilement modifiés pour intégrer nos propositions. Dans notre cas, cela revient à décrire rapidement quelles sont les instructions de ces programmes, quel est l'environnement qui peut être modifié (et dont le contenu définit l'état du programme), et quel est le flot de contrôle général de ces derniers.

Il est important de noter que le type de module d'interprétation de page que nous proposons de transformer (et que nous décrivons ici) est basé sur une recherche descendante, en profondeur d'abord, d'une solution dans l'espace associé. Toutefois, les éléments de syntaxe restent généraux et pourront être adaptés.

6.2.2.1 Syntaxe

Le langage de description de page que nous proposons de considérer est une extension simple, à deux dimensions, de l'exemple de la sous-section 6.2.1. Pour les éléments de langage suivants, nous pourrons distinger deux sortes de constructions :

- les *structures de contrôle*, qui permettent de construire le programme ;
- les *opérateurs*, dont l'utilisation peut avoir des effets de bord sur le programme, et qui peuvent retourner une valeur.

Nous préciserons les types des paramètres et résultats des opérateurs et des expressions lorsque c'est nécessaire, pour le domaine de représentation de la description (le typage des éléments compilés sera différent).

Alternative

Notation : $A \mid B$

Type : sans (structure de contrôle)

Comme dans l'exemple précédent (page 97), cette construction indique la possibilité de reconnaître A ou B.

Concaténation

Notation : $A\,B$

Type : sans (structure de contrôle)

Comme dans l'exemple précédent (page 97), cette construction indique la nécessité de reconnaître A puis B.

Positionnement

Notation : $@P$

Type : Zone $\rightarrow$ ()

Cet *opérateur de positionnement* @ permet de définir une zone de recherche avec la valeur de P pour la suite des calculs, et rappelle la syntaxe des grammaires de position. P est donc une expression dont la valeur est de type Zone.

Élément terminal

Notation : $term(T)$

Type : TypeDonnée $\rightarrow$ DonnéeImage

La modification de l'accès aux éléments terminaux est utile pour récupérer une donnée image (zone et type) d'un type précis contenue dans la zone de recherche. Passer

le type en paramètre de cet opérateur permet de simplifier la gestion du nombre réduit d'éléments primitifs de l'image décrits par leur type : trait, composante connexe, etc.

Dérivation
Notation : $G \longrightarrow D$
Type : sans (structure de contrôle)
Comme dans l'exemple précédent (page 97), cette construction indique l'abstraction de la construction complexe D avec le non-terminal G. Cette construction permet d'indiquer que G est du type de D, si un résultat est retourné.

6.2.2.2 Structures de données

Les structures de données nécessaires au programme d'interprétation sont les suivantes.

Structure image notée I dans la suite, cette structure est un ensemble de couples de type (Zone, TypeDonnée) où les éléments de type TypeDonnée peuvent être « `segment` », « `cercle` », ou encore « `composante_connexe` ». Cette structure est initialisée au lancement du programme avec des détecteurs [9] spécialisés dans la génération de différents symboles, que nous ne présenterons pas.

Zone de recherche notée R dans la suite, cette zone permet de focaliser l'interprétation sur une partie de l'image. Elle est de type Zone mais pourrait être beaucoup plus complexe.

Mémoire Visuelle notée M dans la suite, nous ajoutons d'emblée cette structure dans la sémantique du programme pour simplifier la présentation des opérations sur la mémoire de la sous-section 6.2.3. Cette structure est composée d'un couple de calques (M_T, M_C) contenant chacun des triplets du type (Zone, TypeDonnée, Informations).

Autres structures Le programme de référence que nous proposons de transformer utilise déjà certaines structures de données pour stocker son état. Ces dernières devront être conservées. Nous introduisons donc une structure ε qui portera cette information non spécifiée à ce niveau.

Continuations Afin de permettre une exploration guidée par le but, en profondeur d'abord, de l'espace des solutions, la gestion du retour arrière qui sera représenté par des continuations de succès κ et d'échec ζ dans la suite.

6.2.2.3 Compilation et sémantique du programme

Pour un module d'interprétation de page dont le programme est :
- basé sur une analyse symbolique du contenu ;
- guidé par le but ;
- et explorant l'espace de solution en profondeur d'abord ;

le système de traduction de la description vers les fonctions exécutables peut être spécifié de la façon suivante, largement inspirée de l'exemple de la sous-section 6.2.1.

9. Ces détecteurs sont, à l'instar de l'analyse lexicale pour l'analyse syntaxique en compilation classique, un préalable indispensable à l'interprétation symbolique d'images.

$$
\begin{array}{lcl}
\mathcal{T}[\![A \mid B]\!] & \equiv & \lambda\varepsilon\mathrm{RIM}\kappa\zeta.(\mathcal{T}[\![A]\!]\ \varepsilon\ \mathrm{R}\ \mathrm{I}\ \mathrm{M}\ \kappa\ (\mathcal{T}[\![B]\!]\ \varepsilon\ \mathrm{R}\ \mathrm{I}\ \mathrm{M}\ \kappa\ \zeta)) \\
\mathcal{T}[\![A\ B]\!] & \equiv & \lambda\varepsilon\mathrm{RIM}\kappa\zeta.(\mathcal{T}[\![A]\!]\ \varepsilon\ \mathrm{R}\ \mathrm{I}\ \mathrm{M} \\
 & & \quad \lambda\varepsilon'\mathrm{R}'\mathrm{I}'\mathrm{M}'.(\mathcal{T}[\![B]\!]\ \varepsilon'\ \mathrm{R}'\ \mathrm{I}'\ \mathrm{M}'\ \kappa\ \zeta) \\
 & & \quad \zeta) \\
\mathcal{T}[\![@P]\!] & \equiv & \lambda\varepsilon\mathrm{RIM}\kappa\zeta.(\kappa\ \varepsilon\ [\![P]\!]\ \mathrm{I}\ \mathrm{M}) \\
\mathcal{T}[\![term(T)]\!] & \equiv & \lambda\varepsilon\mathrm{RIM}\kappa\zeta.\begin{cases} (\kappa\ e\ \varepsilon\ \mathrm{R}\ (\mathrm{I}\setminus e)\ \mathrm{M}) & \text{si } (\exists e = (zone, type) \in \mathrm{I} \\ & \quad \wedge type = T \\ & \quad \wedge zone \cap \mathrm{R} \neq \emptyset) \\ \zeta & \text{sinon} \end{cases} \\
\mathcal{T}[\![G]\!] & \equiv & \mathcal{T}[\![D]\!] \quad \text{si } G \longrightarrow D
\end{array}
\tag{6.15}
$$

Par rapport à l'exemple de l'équation 6.13, on peut constater que les règles de réécriture pour l'alternative, la concaténation et la dérivation restent quasiment inchangées : elles se contentent de faire circuler un contexte plus riche. Les modifications les plus importantes sont liées à la gestion du caractère bidimensionnel de la structure à interpréter, à savoir :

1. la possibilité de définir une zone de recherche R avec la valeur de P lors de l'appel à l'*opérateur de positionnement* $@P$;
2. la validation du type et de la position d'un élément terminal, puis sa consommation de la structure image I lors de l'appel à l'*opérateur d'accès aux terminaux* $term(T)$.

On peut remarquer dans les règles de l'équation 6.15 que la circulation des paramètres dépend de leur nature, et on peut en distinguer deux types :

1. ceux qui sont transformés lors des étapes de la reconnaissance, tels que ε, R, I ou M, et se propagent « horizontalement », c'est à dire d'instruction en instruction ;
2. ceux comme κ et ζ, qui sont propagés « verticalement » et construits par les structures de contrôle qui leur donnent une portée, et qui sont utilisés par les opérateurs.

6.2.2.4 Exemple de description

Dans ce type de système, la gestion de variables (déclaration d'identifiants, affectation, et lecture) est relativement simple à mettre en œuvre en gérant un nouveau paramètre qui associerait à chaque identifiant défini (dans le bloc courant) un emplacement mémoire.

On peut alors écrire des descriptions de la forme de l'exemple ci-après, dans lequel on cherche (naïvement) à détecter un rectangle et calculer son aire :

$$
\begin{array}{lcl}
rectangle & \longrightarrow & var\ \text{cote1}\ \text{cote2}\ \text{cote3}\ \text{cote4} \\
 & & @une\_zone \\
 & & \text{cote1} \leftarrow term(\text{trait\_horizal}) \\
 & & @(droite\ \text{cote1}) \\
 & & \text{cote2} \leftarrow term(\text{trait\_vertical}) \\
 & & @(bas\ \text{cote2}) \\
 & & \text{cote3} \leftarrow term(\text{trait\_horizal}) \\
 & & @(gauche\ \text{cote3}) \\
 & & \text{cote4} \leftarrow term(\text{trait\_vertical}) \\
 & & renvoyer\ (calculer\_aire\ \text{cote1}\ \text{cote2}\ \text{cote3}\ \text{cote4})
\end{array}
\tag{6.16}
$$

Dans la description précédente, le concepteur a, bien entendu, dû définir les fonctions *droite*, *bas*, *gauche*, *calculer_aire*, mais on peut supposer que certains éléments soient déjà définis, comme par exemples trait_horizal et trait_horizal du type TypeDonnée.

Ce type de description permet la génération automatique d'un programme d'interprétation d'image à partir d'une description proche d'une grammaire attribuée. Toutefois, ce n'est pas complètement suffisant pour définir un module d'interprétation de page : il faut aussi être capable d'extraire les primitives visuelles de l'image, charger le contenu de la mémoire visuelle en début d'analyse et le sauvegarder à la fin, renvoyer l'état terminal du programme au module de stratégie globale, etc.

Nous montrons rapidement dans la sous-section suivante comment gérer l'intégration de la mémoire visuelle, et ne ferons pas plus de suppositions à propos du fonctionnement d'un éventuel système à adapter.

6.2.3 Formalisation des opérations sur la mémoire visuelle

La gestion du chargement et du déchargement de la mémoire visuelle peut être systématisée en ajoutant une règle de traduction $\mathcal{T}[\![axiome\_mem\ S]\!]$, appliquée une seule fois, pour l'axiome S de la description.

$$\begin{aligned}&\mathcal{T}[\![axiome\_mem\ S]\!]\\ &\quad\equiv \lambda\varepsilon \mathrm{RIM}\kappa\zeta.(\mathcal{T}[\![S]\!]\ \ \varepsilon\ \ \mathrm{R}\ \ \mathrm{I}\ \ (charger\_mem)\\ &\qquad\qquad \lambda\varepsilon'\mathrm{R}'\mathrm{I}'\mathrm{M}'.(sauver\_mem\ \ \mathrm{M}';\ \ \kappa\ \ \varepsilon'\ \ \mathrm{R}'\ \ \mathrm{I}'\ \ \mathrm{M}')\\ &\qquad\qquad \lambda.(sauver\_mem\ \ \mathrm{M};\ \ \zeta))\end{aligned} \tag{6.17}$$

Dans cette règle, on se contente de charger la mémoire avant l'invocation de $\mathcal{T}[\![S]\!]$, et d'indiquer qu'il faudra sauvegarder l'état de la mémoire à la fin, que la reconnaissance réussisse ou échoue. Pour ne pas alourdir la notation, on considère que *charger_mem* renvoie le contenu de la mémoire visuelle, et que *sauver_mem* est une fonction à effet de bord prenant le contenu final de la mémoire en paramètre et l'enregistrant pour le module de stratégie globale. Le point-virgule « ; » indique une séquence d'appels de fonctions où la dernière valeur est renvoyée.

Les opérateurs d'accès à la mémoire définis informellement en sous-section 6.1.2.3 (page 90) peuvent alors être définis précisément. Les types et paramètres des opérateurs sont définis de la façon suivante.

Opérateur *lire_mem* :

TypeDonnée	$\rightarrow$	DonnéeMémoire
type		

Opérateur *consommer_mem* :

TypeDonnée	$\rightarrow$	DonnéeMémoire
type		

Opérateur *ajouter_mem* :

Zone	$\times$	TypeDonnée	$\times$	Informations	$\rightarrow$	DonnéeMémoire
zone		*type*		*info*		

Opérateur *supprimer_mem* :

Zone	$\times$	TypeDonnée	$\times$	Informations	$\rightarrow$	DonnéeMémoire
zone		*type*		*info*		

Les règles de réécriture permettant de passer du langage de description aux fonctions exécutables sont présentées dans les équations ci-après. Le paramètre de mémoire visuelle M est détaillé en $(\mathrm{M_T}, \mathrm{M_C})$ lorsque c'est nécessaire. On pourra remarquer que le premier paramètre passé à la continuation de succès κ correspond au type de retour de l'opérateur dans le langage de description, les autres paramètres représentant l'état du programme.

$$\begin{aligned}
&\mathcal{T}[\![lire_mem(T)]\!] \\
&\quad\equiv\quad \lambda\varepsilon\mathrm{RIM}\kappa\zeta.\begin{cases} (\kappa\ m\ \varepsilon\ \mathrm{R}\ \mathrm{I}\quad \mathrm{M}) & \\ \qquad\qquad\quad \text{si } (\exists m = (zone, type, info) \in \mathrm{M_T} & \\ \qquad\qquad\quad\ \wedge type = T & \\ \qquad\qquad\quad\ \wedge zone \cap \mathrm{R} \neq \emptyset) & \\ \zeta \qquad\qquad \text{sinon} & \end{cases}
\end{aligned} \tag{6.18}$$

$$\begin{aligned}
&\mathcal{T}[\![consommer_mem(T)]\!] \\
&\quad\equiv\quad \lambda\varepsilon\mathrm{RIM}\kappa\zeta.\begin{cases} (\kappa\ m\ \varepsilon\ \mathrm{R}\ \mathrm{I}\quad ((\mathrm{M_T} \setminus m), \mathrm{M_C})) & \\ \qquad\qquad\quad \text{si } (\exists m = (zone, type, info) \in \mathrm{M_T} & \\ \qquad\qquad\quad\ \wedge type = T & \\ \qquad\qquad\quad\ \wedge zone \cap \mathrm{R} \neq \emptyset) & \\ \zeta \qquad\qquad \text{sinon} & \end{cases}
\end{aligned} \tag{6.19}$$

$$\begin{aligned}
&\mathcal{T}[\![ajouter_mem(zone, type, info)]\!] \\
&\quad\equiv\quad \lambda\varepsilon\mathrm{RIM}\kappa\zeta.(\kappa\ (zone, type, info)\ \varepsilon\ \mathrm{R}\ \mathrm{I}\ (\mathrm{M_T}, (\mathrm{M_C} \cup (zone, type, info))))
\end{aligned} \tag{6.20}$$

$$\begin{aligned}
&\mathcal{T}[\![supprimer_mem(F, T, I)]\!] \\
&\quad\equiv\quad \lambda\varepsilon\mathrm{RIM}\kappa\zeta.\begin{cases} (\kappa\ m\ \varepsilon\ \mathrm{R}\ \mathrm{I}\quad ((\mathrm{M_T} \setminus m), (\mathrm{M_C} \cup asupprimer(m))) & \\ \qquad\qquad\quad \text{si } (\exists m = (zone, type, info) \in \mathrm{M_T} & \\ \qquad\qquad\quad\ \wedge type \sim T & \\ \qquad\qquad\quad\ \wedge zone \sim F & \\ \qquad\qquad\quad\ \wedge info \sim I) & \\ \zeta \qquad\qquad \text{sinon} & \end{cases}
\end{aligned} \tag{6.21}$$

Pour la règle de réécriture $\mathcal{T}[\![supprimer\_mem(F, T, I)]\!]$, le symbole $\sim$ correspond à une opération de *matching* permettant d'effectuer une requête de suppression sans connaître précisément la donnée. Pour la règle de réécriture $\mathcal{T}[\![ajouter\_mem(zone, type, info)]\!]$, il serait possible d'améliorer la sémantique donnée en vérifiant, avant l'ajout, que la clé $(zone, type)$ n'est pas déjà utilisée.

Il est alors possible de donner une définition plus précise du fragment de programme associé à la fonction *conserver_resultat* introduite en sous-section 6.1.2.4 (page 92) qui permet de vérifier qu'un résultat de type T est stocké en mémoire avant d'invoquer la règle de description R_{auto}, et qui devient donc un nouvel élément dans la syntaxe de la description. Comme la zone de recherche est devenue un paramètre implicite, la nouveau type de *conserver_resultat* dans le langage de description est le suivant :

$$conserver\_resultat\colon \text{TypeDonnée} \times ((\,) \to (\text{Zone} \times \text{A})) \to (\text{Zone} \times \text{A}) \tag{6.22}$$

$$
\begin{aligned}
&\mathcal{T}[\![conserver\_resultat(T, R_{\text{auto}})]\!] \\
&\quad \equiv \lambda\varepsilon\text{RIM}\kappa\zeta.(\mathcal{T}[\![lire\_mem(T)]\!]\ \varepsilon\ \text{R I M} \\
&\qquad\qquad \lambda m.(\kappa\ (m.zone, m.info)\ \varepsilon\ \text{R I M}) \\
&\qquad\qquad \lambda.(\mathcal{T}[\![R_{\text{auto}}]\!]\ \varepsilon\ \text{R I M} \\
&\qquad\qquad\quad \lambda zi.(\mathcal{T}[\![ajouter\_mem(z, T, i)]\!]\ \varepsilon\ \text{R I M}\ \kappa\ \zeta) \\
&\qquad\qquad\quad \zeta))
\end{aligned}
\tag{6.23}
$$

On utilise alors un marqueur de type T pour indiquer comment stocker (et retrouver) l'information en mémoire le cas échéant.

Au terme de cette section théorique, il est possible de saisir les bases nécessaires à la mise en place d'un mécanisme interactif complexe. La section suivante s'intéresse à la gestion automatique de l'échange d'informations entre le module d'interprétation de page et son environnement grâce à l'expression de nouvelles propriétés à propos du contenu de la page, et se base sur les notations et le module (d'interprétation de page) de référence présentés ici.

6.3 Systématiser les échanges pour corriger les erreurs

À présent que nous avons clairement défini le fonctionnement du module d'interprétation de page que nous souhaitons étendre et la façon de générer son programme par compilation, nous pouvons présenter les nouveaux opérateurs que nous proposons d'intégrer au langage de description de la page pour permettre de gérer automatiquement l'échange asynchrone d'informations entre le module d'interprétation de page et son environnement.

L'objectif de ces nouveaux opérateurs est de permettre l'expression de propriétés utiles qui donneront une structure au mécanisme de détection et correction (asynchrone) d'erreurs au cours de l'exécution du programme d'interprétation d'image.

Nous allons d'abord nous intéresser dans la première sous-section à la façon de gérer automatiquement l'échange d'information lorsque les problèmes peuvent être détectés automatiquement. Nous verrons dans la deuxième sous-section comment garantir une certaine progression du mécanisme global d'interprétation itérative en évitant les boucles et en guidant les réponses à apporter aux problèmes. Dans la troisième section, nous nous intéresserons finalement aux situations où les problèmes ne peuvent pas être détectés automatiquement.

6.3.1 Mise en œuvre de l'interaction dirigée : détection automatique des problèmes

Cette sous-section présente les opérateurs permettant de gérer les cas où la détection automatique de problème est possible, et autorise alors la sollicitation de l'environnement du module d'interprétation de page avec une question précise. Nous présentons ces opérateurs et leur syntaxe, puis nous nous intéressons aux structures de données nécessaires à leur mise en œuvre avant de présenter leur sémantique, et finalement illustrer leur utilisation sur un exemple.

6.3.1.1 Nouveaux opérateurs : description et syntaxe

Les trois nouveaux opérateurs que nous proposons d'ajouter au langage de description de pages visent à mettre en place un mécanisme basé sur le protocole de l'interaction dirigée que nous résumons ci-après.

1. Le module d'interprétation de page tente d'interpréter une image, et pose une question au module de stratégie globale pour chaque problème qu'il détecte, puis termine son travail pour l'itération courante.
2. Lorsque suffisamment de questions et d'informations ont été générées, le module de stratégie globale exploite le contexte documentaire, ou sollicite un opérateur humain, pour répondre aux questions.
3. Lors d'une nouvelle itération, le module d'interprétation de page réintègre les réponses et les utilise pour faire progresser son interprétation.

Afin de mettre en place ce comportement, il est nécessaire de permettre la réalisation de trois actions fondamentales :

1. poser une question lorsqu'on détecte un problème ;
2. continuer l'interprétation dans une partie de l'image indépendante du problème détecté ;
3. utiliser une réponse si elle existe au lieu d'appliquer le raisonnement automatique par défaut pour la région concernée.

Notre approche est alors de considérer ce comportement comme un mécanisme de *détection*, *correction* et *récupération* sur erreur dans lequel la correction est dissociée en une réponse asynchrone faite à l'extérieur du module d'interprétation de page, et une réintégration de la réponse pour corriger l'interprétation lors d'une itération ultérieure. Le mécanisme que nous proposons étant très proche de celui de la gestion d'exception, tant dans son utilisation que dans sa mise en œuvre, nous avons choisi de faire apparaître cette ressemblance dans le nom des opérateurs.

Les trois opérateurs que nous proposons sont alors les suivants :

raise_question(***zone***, ***type***, ***info***)

Type : $(\text{Zone} \times \text{TypeDonnée} \times \text{Informations}) \rightarrow ()$

Cet opérateur peut servir à indiquer une construction incohérente dans le contexte local, l'absence d'un élément indispensable, le manque de confiance dans les résultats d'un classifieur, ou encore que le cas n'est pas géré. C'est au concepteur d'utiliser cet opérateur pour décrire les situations possibles.

Les paramètres de cet opérateur servent à indiquer la position du problème (*zone*), le type d'information manquante (*type*) et d'autres informations (*info*) générées automatiquement pour décrire le contexte de détection du problème.

Lorsque cet opérateur est invoqué, l'analyse s'arrête et reprend à un endroit indépendant du manque d'information constaté, si c'est possible (et décrit à l'aide de l'opérateur *catch_question*).

get_answer_or_try(T, $\boldsymbol{R}_{\textbf{auto}}$)

Type : $(\text{TypeDonnée} \times (() \rightarrow (\text{Zone} \times A)) \rightarrow (\text{Zone} \times A)$

Cet opérateur sert à donner une portée à une correction externe : si une réponse de type T est disponible dans la zone de recherche courante, alors il ne sera pas nécessaire d'appeler la règle de description R_{auto}, et on pourra utiliser l'information plus

fiable fournie par l'environnement. Dans le cas contraire, il faut utiliser le résultat calculé par R_{auto}.
Cet opérateur permet de supprimer une branche de l'arbre d'analyse syntaxique, et d'y substituer la réponse directement fournie par l'environnement du module d'interprétation de page.

catch_question$(T, \mathbf{R_{auto}})$
Type : $(\text{TypeDonnée} \times (() \to (\text{Zone} \times A)) \to \mathit{PeutEtre}(\text{Zone} \times A)$
Cet opérateur sert à indiquer la portée maximale d'un manque d'information d'un type T dans l'analyse, c'est à dire l'indépendance de parties de l'analyse relativement à un problème donné. Il permet de dire que si un problème nécessite l'apport d'information de type T lors de l'exploration de la règle de description R_{auto}, alors il sera possible de reprendre le calcul après l'appel à l'opérateur.
Pour gérer le fait que le résultat du calcul géré par R_{auto} peut ne pas avoir été terminé, le type du résultat de R_{auto} est encapsulé dans le type spécial *PeutEtre* formé de deux constructeurs :

1. `question` qui indique qu'une question a été posée, et qu'aucun résultat n'est disponible ;
2. `auto`(r) qui indique qu'un résultat r produit automatiquement est disponible.

Cet opérateur permet de poser plusieurs questions à chaque étape du processus d'interprétation, en évitant de bloquer sur le premier problème rencontré.

6.3.1.2 Exemple de description

Pour illustrer ce qui vient d'être présenté, considérons le cas simple dans lequel on souhaite détecter et reconnaître les numéros de vente contenus dans la colonne de gauche du tableau de ventes de la figure 6.3. Dans ce premier exemple, nous allons considérer deux types de problèmes :

1. le cas où la colonne contenant les nombres ne peut pas être localisée précisément, à cause de la dégradation des filets, par exemple ;
2. le cas où la valeur d'un nombre reconnue par un classifieur n'est pas fiable.

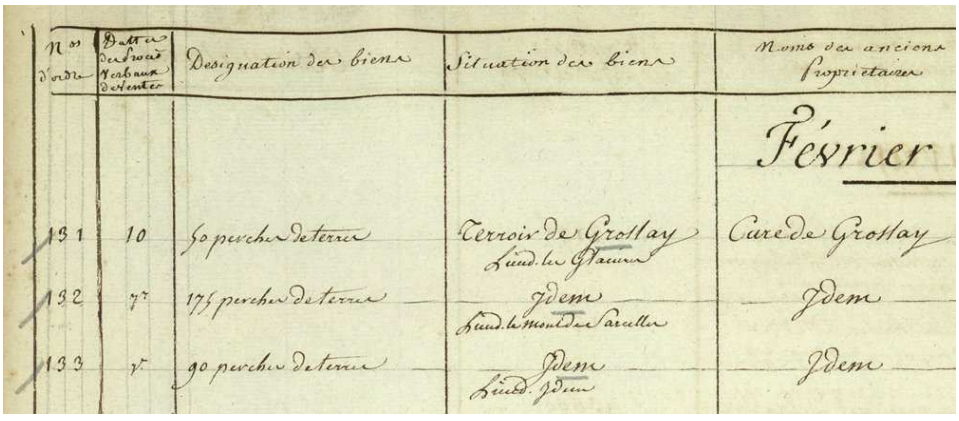


FIGURE 6.3 – Extrait d'un registre de ventes dont on souhaite extraire les numéros de vente (colonne de gauche).

Version sans interaction Une version sans interaction de la description du contenu de la page de la figure 6.3 pourrait donc être la suivante. Notons que, pour alléger la notation, cette description ne produit aucun résultat.

$$\begin{aligned}
&\textit{colonne_no} \longrightarrow \textit{var}\ \ \text{posCol}\\
&\qquad @(\textit{gauchePage})\\
&\qquad (\text{posCol}, _) \leftarrow \textit{localiser_colonne_no}\\
&\qquad @(\text{posCol})\\
&\qquad \textit{localiser_lire_no}\\
&\textit{localiser_lire_no} \longrightarrow \textit{var}\ \ \text{posNo}\ \ \text{posPreciseNo}\ \ \text{ResReco}\\
&\qquad (\text{posNo}, _) \leftarrow \textit{localiser_no}\\
&\qquad @(\text{posNo})\\
&\qquad (\text{posPreciseNo}, \text{ResReco}) \leftarrow \textit{reconnait_no}\\
&\qquad @(\textit{sous}\ \ \text{posNo})\\
&\qquad \textit{localiser_lire_no}\\
&\textit{localiser_lire_no} \longrightarrow \textit{passer}
\end{aligned} \tag{6.24}$$

Quelques précisions sur la notation sont nécessaires.

– On peut considérer que deux dérivations pour la même partie gauche forment une alternative :

$$\mathcal{T}[\![G \longrightarrow D_1\ G \longrightarrow D_2]\!] \equiv \mathcal{T}[\![G \longrightarrow D_1 \mid D_2]\!] \tag{6.25}$$

– La technique utilisée pour détecter et lire tous les nombres est une récursion fonctionnelle classique, où *localiser_lire_no* est rappelée à la fin de son cas général, et où la fonction *passer* termine la récursion si plus aucun nombre ne peut être détecté. La définition de *passer* est triviale :

$$\mathcal{T}[\![\textit{passer}]\!] \equiv \lambda\varepsilon\text{RIM}\kappa\zeta.(\kappa\ \varepsilon\ \text{R}\ \text{I}\ \text{M}) \tag{6.26}$$

– Pour simplifier la définition des règles de description et l'utilisation des opérateurs dans la suite, on considère que les types des règles *localiser_colonne_no*, *localiser_no* et *reconnait_no* sont identiques et équivalents à :

$$() \rightarrow (\text{Zone} \times \text{Informations}) \tag{6.27}$$

et par conséquent il peut être nécessaire d'ignorer (avec un symbole particulier « _ ») la composante de type Informations de la réponse si seule la position est importante, comme c'est le cas pour les règles de localisation.

Dans cet exemple, le concepteur a dû définir les fonctions *localiser_no* et *reconnait_no* selon ses besoins, et éventuellement définir les fonctions *gauchePage* et *sous* si elles n'étaient pas déjà disponibles. Pour permettre de retourner le résultat renvoyé par le classifieur au module de stratégie globale, il devrait également définir globalement (c'est à dire pour tout le système) le type associé. En pratique, il s'agit de structures semblables à des objets (auxquels on peut rajouter des attributs dynamiquement) de la forme suivante :

$$\begin{aligned}
&\{\ \texttt{valeur}\colon 131,\\
&\ \ \texttt{confiance}\colon 0.91,\\
&\ \ \texttt{hypotheses}\colon [(131, 0.91), (137, 0.09)],\\
&\ \ \ldots\ \}
\end{aligned} \tag{6.28}$$

Rappelons que, pour simplifier la présentation, nous avons donné à ces structures associatives le type Informations, qui est le type le plus général de résultat possible.

Dans cet exemple, si un des problèmes identifiés se produit, il est impossible de mettre en attente la suite de l'interprétation pour le corriger. Il est alors nécessaire de faire des suppositions peu fiables qui peuvent entraîner de nouvelles erreurs.

Version avec interaction dirigée Pour palier ce problème d'incertitude dans la version précédente de l'exemple, il est possible d'annoter la description pour indiquer les propriétés utiles à l'aide des opérateurs présentés. La nouvelle description est alors la suivante.

$$
\begin{array}{l}
\textit{colonne\_no} \longrightarrow \\
\quad \textit{var}\ \ \text{posCol} \\
\quad @(\textit{gauchePage}) \\
\quad (\text{posCol}, \_) \leftarrow \textit{get\_answer\_or\_try}(\texttt{colonne\_no\_ordre}, \\
\qquad\qquad \textit{localiser\_colonne\_no}_{int}) \\
\quad @(\text{posCol}) \\
\quad \textit{localiser\_lire\_no} \\
\\
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{var}\ \ \text{posNo}\ \ \text{posPreciseNo}\ \ \text{ResReco} \\
\quad (\text{posNo}, \_) \leftarrow \textit{localiser\_no} \\
\quad @(\text{posNo}) \\
\quad (\text{posPreciseNo}, \text{ResReco}) \leftarrow \textit{catch\_question}(\texttt{no\_vente}, \\
\qquad\qquad \textit{get\_answer\_or\_try}(\texttt{no\_vente}, \textit{reconnait\_no}_{int})) \\
\quad @(\textit{sous}\ \ \text{posNo}) \\
\quad \textit{localiser\_lire\_no} \\
\\
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{passer}
\end{array}
\tag{6.29}
$$

Pour cette description, la concepteur a dû déclarer deux nouveaux éléments appartement au domaine TypeDonnée : `colonne_no_ordre` et `no_vente`. Dans ces deux cas, il faudra aussi adapter les interfaces graphiques si les questions relatives à ces éléments sont présentées à un opérateur humain, afin de déterminer comment les présenter. Nous verrons dans la sous-section 6.3.1.3 comment les questions sont représentées.

La règle de description *localiser_colonne_no*$_{int}$ correspond à une modification mineure de *localiser_colonne_no* pour poser une question lorsqu'il n'est pas possible de trouver la colonne des numéros de vente :

$$
\begin{array}{l}
\textit{localiser\_colonne\_no}_{int} \longrightarrow \\
\quad \textit{localiser\_colonne\_no} \\
\\
\textit{localiser\_colonne\_no}_{int} \longrightarrow \\
\quad \textit{raise\_question}(\text{pageEntiere}, \texttt{colonne\_no\_ordre}, \\
\qquad\qquad \{\texttt{txt}: \text{"Où est la colonne gauche ? "} \\
\qquad\qquad\quad + \text{"Je ne la trouve pas dans "} + \text{R}\}))
\end{array}
\tag{6.30}
$$

où R est la zone de recherche dans laquelle on tente de trouver la colonne.

De la même manière, il est possible de définir [10] une version interactive de *reconnait_no* à partir de cette dernière :

$$
\begin{array}{l}
\textit{reconnait\_no}_{int} \longrightarrow \\
\quad (\text{posPreciseNo}, \text{ResReco}) \leftarrow \textit{reconnait\_no} \\
\quad \textbf{si } (\text{ResReco.confiance} \geq \texttt{SEUIL}) \\
\quad \textbf{alors} \\
\quad\quad \textit{renvoyer } (\text{posPreciseNo}, \text{ResReco}) \\
\quad \textbf{sinon} \\
\quad\quad \textit{raise\_question}(\text{R}, \texttt{no\_vente}, \\
\quad\quad\quad\quad \{\texttt{txt}: \text{"Quelle est la valeur de ce nombre ?"}\}))
\end{array}
\tag{6.31}
$$

Dans la description interactive de l'équation 6.29, un problème de détection de colonne arrêtera l'analyse, car aucun bloc défini par l'opérateur *catch_question* n'englobe l'appel à la règle *localiser_colonne_no*$_{int}$. Cependant, si une réponse satisfaisante (de type `colonne_no_ordre`) est disponible en mémoire, alors la règle *localiser_colonne_no*$_{int}$ ne sera plus appelée car un bloc défini par *get_answer_or_try* protège son appel.

De la même manière, l'échec de la reconnaissance d'un nombre par *reconnait_no*$_{int}$ provoquera la génération d'une question réclamant une information de type `no_vente`. Comme précédemment, un bloc défini par *get_answer_or_try* empêche l'appel de cette règle si une réponse satisfaisante existe au bon endroit en mémoire. Toutefois, comme il est possible de poursuivre l'analyse du reste de la colonne de numéros de vente, un bloc défini par *catch_question* permettra de reprendre le travail en dessous du nombre mal reconnu.

Le version interactive permet donc la mise en place d'une gestion automatique des deux types de problèmes rencontrés, au prix d'une perturbation minime de la description.

6.3.1.3 Mise en œuvre des opérateurs

Avant de décrire le comportement du module d'interprétation de page sur l'exemple précédent, il est nécessaire de présenter la structure des questions et des réponses échangées entre le module d'interprétation de page et son environnement. Nous complétons la présentation de la mise en œuvre (abstraite) des opérateurs par la description d'une première sémantique possible.

Structures de données nécessaires Au niveau de la gestion de la mémoire et des données échangées, il est nécessaire de pouvoir stocker questions et réponses. Pour les questions, il est nécessaire d'ajouter un nouvel élément « `question` » au domaine TypeDonnée. La structure d'une question est un élément de type DonnéeMémoire de la forme suivante :

$$
(\text{zone}, \texttt{question}, \{\texttt{txt}: Q, \texttt{type\_attendu}: T, \ldots\}) \tag{6.32}
$$

où Q représente une question sous forme textuelle générée automatiquement en fonction du contexte local du module d'interprétation de page, selon ce que le concepteur a souhaité indiquer, et $T \in$ TypeDonnée représente un type défini par le concepteur. D'autres éléments peuvent être stockés dans les informations de la question, comme des hypothèses de reconnaissance par exemple, qui peuvent être nécessaires pour apporter une réponse à la question. Ces informations dépendent donc du type de donnée qui doit être fournie, et à priori seuls les éléments `txt` et `type_attendu` sont indispensables.

10. La mise en œuvre d'une structure de contrôle conditionnelle est immédiate.

Concernant les réponses, il est nécessaire de garder une grande homogénéité avec les informations provenant de l'image afin de permettre une fusion facile. Une réponse à la question précédente est donc un élément de type DonnéeMémoire de la forme suivante :

$$(\text{zonePrecise}, T, \{\ \texttt{valeur}\colon \mathit{Val}, \ldots\ \}) \tag{6.33}$$

où *Val* est, à minima, la valeur fournie par l'environnement.

Outre la gestion des questions et des réponses, il est nécessaire de permettre au programme d'interprétation de reprendre son travail après qu'une question soit posée, afin d'éviter que ce dernier reste bloqué sur le premier problème rencontré s'il est possible de poursuivre l'analyse sur d'autres parties de la page. Ce mécanisme de récupération sur erreur, géré au sein des opérateurs *catch_question* et *get_answer_or_try*, est semblable à une gestion d'exceptions.

Il est donc nécessaire d'ajouter à la sémantique du programme un paramètre E qui circule comme ε (sauf au niveau du catch). C'est une liste de couples du type

$$\text{TypeDonnée} \times \text{Continuation} \tag{6.34}$$

qui stocke, en fonction de l'information attendue dans une question (ce qui caractérise le problème), le point de reprise, ou *continuation d'interaction*, à invoquer après avoir enregistré la question en mémoire. Cette structure de donnée représente l'empilement des blocs *catch_question* qui délimitent la portée de chaque type d'erreur. Pour gérer le cas où aucun bloc *catch_question* ne correspond pas à une question posée, il faut initialiser E au début du processus d'interprétation pour indiquer une continuation par défaut. Nous utilisons le caractère « _ » dans la nouvelle règle de réécriture de l'axiome de la description suivante, où E = [ ] (liste vide) par défaut :

$$\mathcal{T}[\![\mathit{axiome\_int}\ S]\!] \quad \equiv \quad \mathcal{T}[\![\mathit{axiome\_mem}\ (\mathit{catch\_question}(\_, S))]\!] \tag{6.35}$$

Sémantique et règles de compilation Il est à présent possible de présenter une première version des règles de compilation des opérateurs.

$$\mathcal{T}[\![\mathit{get\_answer\_or\_try}(T, R_{\text{auto}})]\!] \quad \equiv \quad \mathcal{T}[\![\mathit{conserver\_resultat}(T, R_{\text{auto}})]\!] \tag{6.36}$$

$$\begin{aligned}
&\mathcal{T}[\![\mathit{catch_question}(T, R_{\text{auto}})]\!] \\
&\quad\equiv\quad \lambda\varepsilon\mathrm{RIME}\kappa\zeta.(\mathcal{T}[\![R_{\text{auto}}]\!]\ \varepsilon\ \mathrm{R}\ \mathrm{I}\ \mathrm{M} \\
&\qquad\qquad (\mathit{cons}(\,(T, \lambda\varepsilon'\mathrm{R}'\mathrm{I}'\mathrm{M}'\mathrm{E}'.(\kappa\ \texttt{question}\ \varepsilon'\ \mathrm{R}'\ \mathrm{I}'\ \mathrm{M}'\ \mathrm{E})), \\
&\qquad\qquad\quad E)\,) \\
&\qquad\qquad \lambda r\varepsilon'\mathrm{R}'\mathrm{I}'\mathrm{M}'\mathrm{E}'.(\kappa\ \texttt{auto}(r)\ \varepsilon'\ \mathrm{R}'\ \mathrm{I}'\ \mathrm{M}'\ \mathrm{E}) \\
&\qquad\qquad \zeta)
\end{aligned} \tag{6.37}$$

$$
\begin{aligned}
&\mathcal{T}[\![\mathit{raise\_question}(\mathit{zone}, \mathit{type}, \mathit{info})]\!] \\
&\quad \equiv \quad \lambda\varepsilon\mathrm{RIME}\kappa\zeta.(\mathcal{T}[\![\mathit{ajouter\_mem}(\mathit{zone}, \texttt{question}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad \{\texttt{type\_attendu}\colon \mathit{type}\} \cup \mathit{info})]\!] \\
&\qquad\qquad\qquad\qquad\qquad\qquad \varepsilon\ \mathrm{R}\ \mathrm{I}\ \mathrm{M}\ \mathrm{E} \\
&\qquad\qquad\qquad\qquad\qquad\qquad \lambda m.(C) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \zeta) \\
&\qquad \text{avec } C = \mathit{chercher}(\mathit{type}, \mathrm{E}) \\
&\qquad \text{et où } \mathit{chercher}(\mathit{type}, \mathrm{E}) \text{ retourne le premier } C = (T, C) \in \mathrm{E} \text{ (le plus récent)} \\
&\qquad\quad \text{tel que } (T = \mathit{type} \vee T = \_)
\end{aligned}
\tag{6.38}
$$

Cette première version de nos opérateurs montre qu'on peut se baser avantageusement sur l'opérateur *conserver_resultat* pour construire directement la fonction associée à *get_answer_or_try*. La gestion d'exception implémentée ici est différente de celle qu'on peut trouver habituellement dans les langages de programmation impérative, et dans lesquels le programme remonte bloc par bloc en vérifiant à chaque fois si le type géré est celui de l'exception lancée.

Ici, le point de reprise du programme est déterminé lors de l'appel à *raise_question* en cherchant dans la pile de *continuations d'interaction*. Cette pile de *continuations d'interaction* est enrichie par l'opérateur *catch_question* lors de son invocation : il passe à la règle de description R_{auto} une valeur de la pile E mise à jour, et dans le cas d'une question ou d'une analyse automatique qui aboutit, la continuation construite prend soin de réinitialiser la valeur de E pour définir correctement une portée au bloc défini par l'opérateur *catch_question*. Afin de distinguer le cas où R_{auto} termine de celui où une question est posée, *catch_question* passe à sa continuation de succès κ un paramètre supplémentaire `question` ou `auto`, selon le cas, afin de permettre, si nécessaire, la mise en place d'un résultat par défaut.

Le texte (généré dynamiquement) de la question est passé dans le paramètre *info* lors de l'appel à *raise_question*.

6.3.1.4 Exemple de comportement du module d'interprétation de page

Pour illustrer le comportement du module d'interprétation de page, nous nous basons sur l'exemple de localisation et de reconnaissance de numéros de ventes présenté précédemment en sous-section 6.3.1.2, page 106. Nous allons nous donner une stratégie globale minimaliste qui va nous permettre d'observer l'évolution de la mémoire visuelle associée à *une* page au cours du traitement d'un ensemble de pages. L'évolution du contenu de la mémoire visuelle, que nous décrirons, traduit bien le comportement du programme d'interprétation.

Stratégie globale Pour imaginer le comportement du module d'interprétation de page, il est nécessaire d'intégrer ce dernier à un système complet capable d'invoquer ce module, et de lui fournir les données qu'il sollicite. L'algorithme 2 donne l'idée générale d'une stratégie globale naïve traitant les pages séparément. Elle va nous permettre de nous focaliser sur une page, sans tenir compte de l'interprétation des autres pages, pour simplifier l'explication. *BDC* représente la base de donnée centrale, *MIP* le module d'interprétation de page, et *IHM* une interface homme–machine permettant de communiquer avec un opérateur humain. Une page sera considérée comme complètement traitée lorsqu'il n'y aura plus

de question dans la mémoire visuelle qui lui est associée. Notons que cette stratégie ne tire absolument pas profit des possibilités d'interaction asynchrone entre les modules, mais elle a pour objectif de permettre une explication concise.

Algorithm 2 Exemple de stratégie globale pour l'interaction dirigée

```
1:  procedure TRAITER_LOT(lot)
2:      while lot ≠ ∅ do
3:          p ← lot.extraire()
4:          p.cpt ← p.cpt + 1                ▷ p.cpt compte le nombre de passes sur p
5:          m ← BDC.extraire_memoire(p)
6:          (r, Δm) ← MIP.interpreter(p, m)
7:          m ← fusionner(m, Δm)                                   ▷ point (A)
8:          if contient_question(m) then
9:              Δm ← IHM.poser_questions(p, m)
10:             m ← fusionner(m, Δm)                               ▷ point (B)
11:             lot.ajouter(p)                  ▷ Il faut réinterpréter p
12:         end if
13:         DBC.stocker_memoire(m)
14:     end while
15: end procedure
```

Évolution du contenu de la mémoire visuelle associée à l'image Nous allons nous intéresser au contenu de la mémoire visuelle m associée à la page p après l'exécution des instructions aux points (A) (après la réponse du module d'interprétation de page) et (B) (après la réponse d'un opérateur humain) dans l'algorithme 2. Nous allons décrire l'évolution de son contenu au cours des différentes passes d'interprétation sur l'image, en expliquant comment les données ont été produites. Nous allons supposer que la page que nous suivons (celle de la figure 6.3 page 106) est assez effacée, et que la détection de la colonne pose problème, ainsi que la lecture du nombre de la ligne du milieu (« 132 »).

Contenu de la mémoire visuelle *Au lancement du traitement, avant l'itération 1*

{ }

Observations : Le traitement vient de commencer, la mémoire visuelle associée à la page est vide.

Contenu de la mémoire visuelle *Itération 1, point (A)*

{ (pageEntiere, `question`, { `txt`: "Où est la colonne [...]",
`type_attendu`: `colonne_no_ordre` }) }

Observations : Le module d'interprétation de page n'a pas pu détecter automatiquement la position de la colonne des numéros de vente. Il a généré une question pour obtenir cette information.

Contenu de la mémoire visuelle *Itération 1, point* (B)

{ (zoneColonne, `colonne_no_ordre`, { }) }

Observations : Un opérateur humain a indiqué la position zoneColonne de la colonne contenant les numéros de vente. Ceci a provoqué la suppression de la question associée.

Contenu de la mémoire visuelle *Itération 2, point* (A)

{ (zoneColonne, `colonne_no_ordre`, { }),
(zoneNo1, `no_vente`, { `valeur: 131` }),
(zoneNo2, `question`, { `type_attendu: no_vente`,
`txt:` "Quelle est la valeur de ce nombre ?" }),
(zoneNo3, `no_vente`, { `valeur: 133` }) }

Observations : Le module d'interprétation de page a localisé trois nombres, mais n'en a reconnu que deux. Il a posé une question pour connaître la valeur du nombre inconnu.

Contenu de la mémoire visuelle *Itération 2, point* (B)

{ (zoneColonne, `colonne_no_ordre`, { }),
(zoneNo1, `no_vente`, { `valeur: 131` }),
(zoneNo2, `no_vente`, { `valeur: 132` }),
(zoneNo3, `no_vente`, { `valeur: 133` }) }

Observations : Un opérateur humain a indiqué la valeur du nombre inconnu. La question associée a été supprimée.

Contenu de la mémoire visuelle *Itération 3, point* (A)

{ (zoneColonne, `colonne_no_ordre`, { }),
(zoneNo1, `no_vente`, { `valeur: 131` }),
(zoneNo2, `no_vente`, { `valeur: 132` }),
(zoneNo3, `no_vente`, { `valeur: 133` }) }

Observations : Un dernier passage par le module d'interprétation de page est nécessaire pour réintégrer les nouvelles informations. Il termine le traitement de la page.

Dans cette sous-section, nous avons vu comment modifier le comportement automatique du module d'interprétation de page grâce aux éléments que nous avons mis en place, lorsqu'il est possible de détecter automatiquement les problèmes se produisant. Avant de nous intéresser (dans la sous-section 6.3.3) au cas où la détection automatique n'est pas possible, il faut noter dans les propositions qui viennent d'être faites, pour la mise en œuvre de l'interaction dirigée, deux manquements qui empêchent de garantir (dans une certaine mesure) une progression de l'interprétation.

Au niveau du positionnement des réponses, tout d'abord. Bien que le module de stratégie globale reçoive du module d'interprétation de page une question précise quant au domaine dans lequel il doit choisir la valeur de la réponse, il n'en est pas de même pour le positionnement de cette dernière. Est-il possible de déplacer la réponse par rapport à

la question ? Dans quelles limites ? Il est nécessaire de clarifier ces éléments de positionnement pour garantir que la réponse sera bien détectée dans une des zones de recherche utilisées.

Au niveau des éventuelles erreurs de conceptions, il faut savoir qu'en pratique l'ordre d'imbrication des opérateurs d'interaction a une certaine importance : pour un type de question donné, l'appel à l'opérateur *raise_question* doit être contenu dans un bloc *get_answer_or_try* lui-même contenu dans un bloc *catch_question*. Une erreur dans la position du bloc *catch_question* n'est pas très gênante en pratique, car l'initialisation automatique de la description permettra au programme de terminer correctement, et le concepteur se privera simplement de la possibilité de générer plusieurs questions au cours d'une itération. Dans le cas de l'imbrication entre *raise_question* et *get_answer_or_try*, le problème est plus sérieux, car si la réponse à une question générée par *raise_question* n'est pas utilisée par un bloc *get_answer_or_try* englobant, alors *la question sera posée indéfiniment, même si la réponse existe,* rien ne permettra de couper l'appel à la règle de description automatique. Il est donc nécessaire de faciliter la détection de ces erreurs de conception pour éviter la formation de boucles infinies dans le mécanisme d'interprétation itératif global.

La sous-section suivante montre comment régler ces deux problèmes.

6.3.2 Garantir une progression de l'interprétation : valider questions et réponses

L'objectif de cette sous-section est double :

1. montrer qu'il est possible de modifier facilement la mise en œuvre des opérateurs proposés pour améliorer le programme d'interprétation généré, sans changer le sens de la description de la page ;
2. régler, au moins partiellement, les problèmes de progression mentionnés dans la section précédente, en permettant une validation des questions et des réponses.

Nous allons donc montrer comment garantir qu'une question ne sera pas posée à nouveau si une réponse satisfaisante existe en mémoire. Nous allons également montrer comment générer automatiquement des contraintes de positionnement des réponses, afin de guider les opérateurs humains dans le positionnement de ces dernières et garantir leur utilité.

6.3.2.1 Syntaxe

La syntaxe des opérateurs reste identique, la façon de décrire une page ne change pas.

6.3.2.2 Structures de données nécessaires

Afin de permettre de gérer le risque de boucle identifié précédemment, nous distinguons deux types de questions :

- les *questions valides*, pour lesquelles un bloc *get_answer_or_try* recherchant un élément du bon type T englobe l'appel à l'opérateur *raise_question*,
- et les *questions invalides*, pour lesquelles ce n'est pas le cas.

Pour savoir, lors de la génération d'une question, si celle-ci est correcte ou non, il est nécessaire de connaître l'arbre d'appel dynamique des fonctions menant à l'invocation de *raise_question* Une certaine détection statique (à la compilation de la description) serait possible, mais elle ne pourrait pas tenir compte des éventuelles constructions contextuelles

que le langage peut décrire, car il faudrait explorer toutes les dérivations syntaxiques possibles, ce qui n'est pas envisageable. Une détection à l'exécution nous a semblé, en pratique, un bon compromis.

Nouveau paramètre pour les instructions L'intégration d'un nouveau paramètre V (pour « validité ») pour les instructions du programme est alors nécessaire. Ce paramètre circule classiquement, c'est à dire comme ε, sauf au niveau de l'opérateur *get_answer_or_try* qui le modifie, et au niveau de l'opérateur *raise_question* qui l'utilise. Il contient l'ensemble des types pour lesquels il est possible de poser une question.

Afin de faciliter la production d'une réponse utile par l'environnement, il est intéressant de stocker pour chaque type T la zone R servant à rechercher une éventuelle réponse en mémoire au niveau de l'opérateur *get_answer_or_try*. Ceci permet de guider l'environnement du module d'interprétation de page, et plus particulièrement les interfaces graphiques, dans le contrôle des réponses qui peuvent être acceptées.

Par conséquent, le nouveau paramètre V contient un ensemble (initialement vide) de couples du type

$$\text{TypeDonnée} \times \text{Zone}. \tag{6.39}$$

Nouvelles zones Cette connaissance de la zone dans laquelle sera recherchée la réponse peut être intégrée à la question pour guider la réponse externe. Les questions contiennent alors une nouvelle information `zone_acceptation` qui correspond à la zone dans laquelle une réponse peut être acceptée. Pour chaque couple question–réponse, trois zones sont alors nécessaires :

La zone de la question notée zoneQuestion ci-après, elle localise le problème.

La zone de recherche de la réponse dite *zone d'acceptation* et notée zoneAcceptation ci-après, elle correspond à la zone de recherche au début du bloc défini par *get_answer_or_try* englobant l'appel à *raise_question*.

La zone de la réponse notée zoneReponse ci-après, elle localise la réponse précisément.

Lors de la création d'une réponse, le module ou l'interface graphique utilisé doit garantir que $\text{zoneReponse} \cap \text{zoneAcceptation} \neq \emptyset$. Par ailleurs, on peut faciliter la saisie d'une réponse par un opérateur humain en initialisant zoneReponse avec la valeur de zoneQuestion.

Représentation des questions et des réponses Si la question est valide, c'est à dire qu'il existe un bloc *get_answer_or_try* recherchant un élément du bon type T et englobant l'appel à l'opérateur *raise_question*, alors elle présente de la forme suivante :

$$\begin{aligned}(\text{zoneQuestion}, \texttt{question}, \{\ &\texttt{txt}: Q,\\ &\texttt{type\_attendu}: T,\\ &\texttt{zone\_acceptation}: \text{zoneAcceptation},\\ &\dots\ \})\end{aligned} \tag{6.40}$$

et la forme de la réponse associée est alors :

$$(\text{zoneReponse}, T, \{\ \texttt{valeur}: \mathit{Val}, \dots\ \}) \tag{6.41}$$

Si, au contraire, la question est invalide, c'est à dire qu'aucun bloc *get_answer_or_try* ne permet d'éviter d'appeler indéfiniment l'opérateur *raise_question*, alors elle a la forme

suivante :

$$(\text{zoneQuestion}, \texttt{question\_invalide}, \{\, \texttt{txt}: Q, \quad \texttt{type\_attendu}: T, \quad \ldots\,\}) \tag{6.42}$$

L'intérêt de l'export de contraintes pour les réponses vers les autres modules est d'éviter de dupliquer la connaissance à propos du contenu des pages lors de la conception, et de transmettre automatiquement les informations nécessaires lors de l'exécution.

6.3.2.3 Compilation et sémantique

La séparation stricte entre le langage de description de la page et l'implémentation des opérateurs autorise à changer l'implémentation de ces derniers sans avoir d'impact sur la description, et donc sur le travail du concepteur.

La gestion du nouveau paramètre V et la distinction des questions valides et invalides nécessite une nouvelle version des règles de réécriture associées aux opérateurs.

$$\begin{aligned}
&\mathcal{T}[\![\mathit{get_answer_or_try}(T, R_{\text{auto}})]\!] \\
&\quad \equiv \quad \lambda\varepsilon\text{RIMEV}\kappa\zeta.(\mathcal{T}[\![\mathit{conserver_resultat}(T, R_{\text{auto}})]\!] \\
&\qquad\qquad \varepsilon\ \text{R}\ \text{I}\ \text{M}\ \text{E} \\
&\qquad\qquad (\mathit{cons}(\,(T, \text{R}), \text{V})\,) \\
&\qquad\qquad \lambda r\varepsilon'\text{R}'\text{I}'\text{M}'\text{E}'\text{V}'.(\kappa\ r\ \varepsilon'\ \text{R}'\ \text{I}'\ \text{M}'\ \text{E}'\ \text{V}) \\
&\qquad\qquad \zeta)
\end{aligned} \tag{6.43}$$

L'opérateur *get_answer_or_try* est responsable de la création d'un couple (*type*, *zone*) et lui donne une portée en supprimant cet élément en sortie de bloc (en rétablissant la valeur de V, reçue à l'entrée du bloc, lors de l'appel à κ).

$$\begin{aligned}
&\mathcal{T}[\![\mathit{catch_question}(T, R_{\text{auto}})]\!] \\
&\quad \equiv \quad \lambda\varepsilon\text{RIMEV}\kappa\zeta.(\mathcal{T}[\![R_{\text{auto}}]\!]\ \varepsilon\ \text{R}\ \text{I}\ \text{M} \\
&\qquad\qquad (\mathit{cons}(\,(T, \lambda\varepsilon'\text{R}'\text{I}'\text{M}'\text{E}'\text{V}'.(\kappa\ \texttt{question}\ \varepsilon'\ \text{R}'\ \text{I}'\ \text{M}'\ \text{E}\ \text{V})), \\
&\qquad\qquad\quad E)\,) \\
&\qquad\qquad \lambda r\varepsilon'\text{R}'\text{I}'\text{M}'\text{E}'\text{V}'.(\kappa\ \texttt{auto}(r)\ \varepsilon'\ \text{R}'\ \text{I}'\ \text{M}'\ \text{E}\ \text{V}') \\
&\qquad\qquad \zeta)
\end{aligned} \tag{6.44}$$

Pour l'opérateur *catch_question*, la modification consiste simplement à rétablir la bonne valeur de V si une question est posée.

$$
\begin{aligned}
&\mathcal{T}[\![\mathit{raise\_question}(\mathit{zone}, \mathit{type}, \mathit{info})]\!] \\
&\equiv \quad \lambda\varepsilon\mathrm{RIMEV}\kappa\zeta. \begin{cases}
(\mathcal{T}[\![\mathit{ajouter\_mem}(\mathit{zone}, \texttt{question}, \\
\qquad\qquad \{\texttt{type\_attendu}: \mathit{type}, \\
\qquad\qquad\ \texttt{zone\_acceptation}: R_{\text{try}}\} \cup \mathit{info})]\!] \\
\qquad \varepsilon\ \mathrm{R\ I\ M\ E\ V}\ \ \mathit{chercher}(\mathit{type}, \mathrm{E})\ \ \zeta) \\
\qquad\qquad \text{si } \exists (T_{\text{try}}, R_{\text{try}}) \in \mathrm{V} \wedge \mathit{type} = T_{\text{try}} \\
(\mathcal{T}[\![\mathit{ajouter\_mem}(\mathit{zone}, \texttt{question\_invalide}, \\
\qquad\qquad \{\texttt{type\_attendu}: \mathit{type}\} \cup \mathit{info})]\!] \\
\qquad \varepsilon\ \mathrm{R\ I\ M\ E\ V}\ \ \mathit{chercher}(\_, \mathrm{E})\ \ \zeta) \\
\qquad\qquad \text{sinon}
\end{cases}
\end{aligned}
\tag{6.45}
$$

L'opérateur *raise_question* reprend la mise en œuvre précédente en ajoutant un test supplémentaire pour savoir si la question est valide ou non, et le cas échéant indiquer la zone d'acceptation de la réponse.

6.3.2.4 Exemple

Nous reprenons ici l'exemple, qui sert de cadre à cette section, visant à localiser puis à reconnaître des numéros de vente dans la colonne de gauche de tableaux (voir figure 6.4).

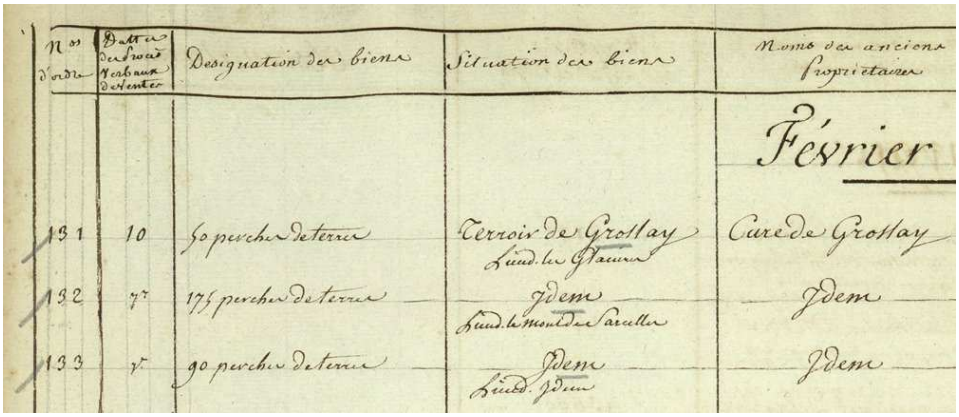


FIGURE 6.4 – Extrait d'un registre de ventes dont on souhaite extraire les numéros de vente (colonne de gauche). *Rappel de la figure 6.3, page 106.*

Pour cet exemple, on ne considérera pas que la détection de la colonne puisse poser problème, et on s'intéressera simplement à la détection et à la reconnaissance des nombres.

Cas avec une question valide Le fragment de description intéressant ne concerne alors que la règle de description *localiser_lire_no* dans l'exemple précédent, qu'on reprend dans l'équation 6.46 suivante.

$$
\begin{array}{l}
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{var}\ \ \text{posNo}\ \ \text{posPreciseNo}\ \ \text{ResReco} \\
\quad (\text{posNo}, \_) \leftarrow \textit{localiser\_no} \\
\quad @(\text{posNo}) \\
\quad (\text{posPreciseNo}, \text{ResReco}) \leftarrow \textit{catch\_question}(\texttt{no\_vente}, \\
\qquad\qquad\qquad\qquad \textit{get\_answer\_or\_try}(\texttt{no\_vente}, \textit{reconnait\_no}_{int})) \\
\quad @(\textit{sous}\ \ \text{posNo}) \\
\quad \textit{localiser\_lire\_no} \\
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{passer}
\end{array}
\tag{6.46}
$$

En reprenant la même stratégie globale que précédemment (voir l'algorithme 2, page 112), l'évolution du contenu de la mémoire visuelle au moment intéressant du traitement se fera comme suit :

Contenu de la mémoire visuelle *Itération 1, point* (A)

```
{ (zoneNo1, no_vente, { valeur: 131 }),
  (zoneNo2, question, { type_attendu: no_vente,
                        zone_acceptation: zoneColonne,
                        txt: "Quelle est la valeur de ce nombre ?" }),
  (zoneNo3, no_vente, { valeur: 133 }) }
```

Observations : Le module d'interprétation de page a localisé trois nombres, mais n'en a reconnu que deux. Il a posé une question pour connaître la valeur du nombre inconnu *en indiquant la zone dans laquelle le résultat doit se trouver.*

Contenu de la mémoire visuelle *Itération 1, point* (B)

```
{ (zoneColonne, colonne_no_ordre, { }),
  (zoneNo1, no_vente, { valeur: 131 }),
  (zoneNo2Precise, no_vente, { valeur: 132 }),
  (zoneNo3, no_vente, { valeur: 133 }) }
```

Observations : Un opérateur humain a indiqué la valeur du nombre inconnu *en précisant la position du nombre*. La question associée a été supprimée.

Cas avec une question invalide Un autre cas intéressant est celui où une erreur est présente dans la description : aucun bloc *get_answer_or_try* n'est défini autour de l'appel à $\textit{reconnait\_no}_{int}$ qui peut poser une question.

$$
\begin{array}{l}
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{var}\ \ \text{posNo posPreciseNo ResReco} \\
\quad (\text{posNo}, \_) \leftarrow \textit{localiser\_no} \\
\quad @(\text{posNo}) \\
\quad (\text{posPreciseNo}, \text{ResReco}) \leftarrow \textit{catch\_question}(\texttt{no\_vente}, \textit{reconnait\_no}_{\textit{int}}) \\
\quad @(\textit{sous}\ \ \text{posNo}) \\
\quad \textit{localiser\_lire\_no} \\
\textit{localiser\_lire\_no} \longrightarrow \\
\quad \textit{passer}
\end{array}
\tag{6.47}
$$

Dans ce cas, la première question provoquera un arrêt de l'interprétation et la notification du problème.

Contenu de la mémoire visuelle *Itération 1, point* (A)

```
{ (zoneNo1, no_vente, { valeur: 131 }),
  (zoneNo2, question_invalide, { type_attendu: no_vente,
                                 txt: "Quelle est la valeur [...]" }) }
```

Observations : Le module d'interprétation de page a localisé un nombre, et a tenté de poser une question pour réclamer la valeur du second nombre. Le risque de boucle infinie a été détecté et l'interprétation de la page courante arrêtée.

Au delà de l'intérêt pour limiter les erreurs de conception, cette amélioration de la mise en œuvre des opérateurs montre qu'il est facile de les adapter sans changer le sens de la description du contenu de la page. La formalisation des opérateurs permet de raisonner rigoureusement à propos du comportement du programme généré, et l'export de contraintes de position ou de domaine de réponse vers les autres modules permet d'éviter la duplication de connaissances relatives au contenu des pages lors de la conception pour laisser le système gérer automatiquement l'échange d'informations importantes lors de l'exécution.

6.3.3 Intégration homogène de l'interaction spontanée : gérer les problèmes non détectés

Jusqu'à présent, les opérateurs de description proposés permettent de mettre en place une interaction *dirigée*, c'est à dire basée sur la détection automatique de problèmes et la réponse ciblée à l'extérieur du module d'interprétation de page. Pour faire face aux cas où la détection automatique de problème n'est pas possible, il est nécessaire de permettre la mise en place d'une interaction *dirigée* qui autorise la détection *et* la correction de problèmes à l'extérieur du module d'interprétation de page.

Les raisons pour lesquelles certains problèmes ne peuvent pas être détectés automatiquement peuvent être multiples, et on peut citer :

- le manque de connaissance à propos du contenu réel des document d'un fonds ;
- la difficulté à détecter un problème et à le localiser précisément ;
- l'absence d'implémentation du mécanisme de détection, par manque de temps ou de moyens lors de la conception du système.

Nous proposons ici un nouvel opérateur capable d'exprimer l'incertitude d'une partie de la description. Il autorise les modifications *spontanées*, c'est à dire multiples et optionnelles, des résultats hors du module d'interprétation de page qui réintégrera ces nouvelles données. Cet opérateur doit être couplé à l'utilisation de l'opérateur *get_answer_or_try* pour permettre l'exploitation des réponses. Il est parfaitement compatible avec le mécanisme d'interaction *dirigée* qui peut être utilisé simultanément.

6.3.3.1 Syntaxe du nouvel opérateur

L'opérateur que nous proposons délimite une portion de description incertaine pour laquelle des éléments pourront être remplacés ou ajoutés grâce à une interprétation externe intégrant plus de connaissances et de contexte.

Concernant le choix du nom de l'opérateur, nous avons pensé qu'il était judicieux d'avoir un nom assez générique pour pouvoir représenter les différentes situations dans lesquelles il est utile d'exprimer la possibilité de modifications externes. Le nom et le type de cet opérateur sont donc les suivants :

$spontaneous(T, \mathbf{R_{auto}})$
Type : $(\text{TypeDonnée} \times (() \to (\text{Zone} \times A)) \to (\text{Zone} \times A)$

De même que pour les autres opérateurs, c'est au concepteur de définir, dans la description de la page, quand il souhaite utiliser *spontaneous*.

6.3.3.2 Structures de données nécessaires

Pour indiquer la possibilité d'apporter zéro, une ou plusieurs informations d'un type donné à un endroit précis dans l'image, nous proposons d'introduire une nouvelle sorte de question *optionnelle* (toujours de type DonnéeMémoire) de la forme suivante :

$$\begin{aligned}(\text{zoneQuestion}, \texttt{question\_optionnelle}, \{\ &\texttt{txt}: Q,\\ &\texttt{type\_attendu}: T,\\ &\texttt{zone\_acceptation}: \text{zoneAcceptation},\\ &\ldots\ \})\end{aligned} \tag{6.48}$$

Cette question pourra accepter aucune, une ou plusieurs réponses de la forme classique suivante :

$$(\text{zonePrecise}, T, \{\ \texttt{valeur}: Val, \ldots\ \}) \tag{6.49}$$

où $\text{zonePrecise} \cap \text{zoneAcceptation} \neq \emptyset$.

6.3.3.3 Compilation et sémantique

Si on considère qu'il est possible d'utiliser la gestion de variables dont nous nous sommes servis pour simplifier l'écriture des descriptions précédentes, alors cet opérateur

peut être mis en œuvre de la façon suivante, directement dans le langage de description :

$$\begin{aligned} spontaneous(T, R_{\text{auto}}) \longrightarrow\ & var\ \ z_R \\ & z_R \leftarrow acces\_zone\_recherche \\ & ajouter\_mem(z_R, \texttt{question\_optionnelle}, \\ & \qquad \{\, \texttt{type\_attendu:}\ T \,\}) \\ & R_{\text{auto}} \end{aligned} \tag{6.50}$$

L'opérateur pose systématiquement une question optionnelle, et appelle la règle de description R_{auto}.

Pour simplifier la description, nous avons introduit un nouvel opérateur *acces_zone_recherche* qui renvoie la zone de recherche courante. Son type est :

$$() \rightarrow \text{Zone} \tag{6.51}$$

et sa définition est :

$$\mathcal{T}[\![acces\_zone\_recherche]\!] \equiv \lambda\varepsilon\text{RIMEV}\kappa\zeta.(\kappa\ \text{R}\ \varepsilon\ \text{R}\ \text{I}\ \text{M}\ \text{E}\ \text{V}) \tag{6.52}$$

6.3.3.4 Exemple

Pour illustrer l'intérêt de ce nouvel opérateur, on reprend l'exemple des numéros d'ordre contenus dans la colonne de gauche des tableaux tels que celui rappelé dans la figure 6.5.

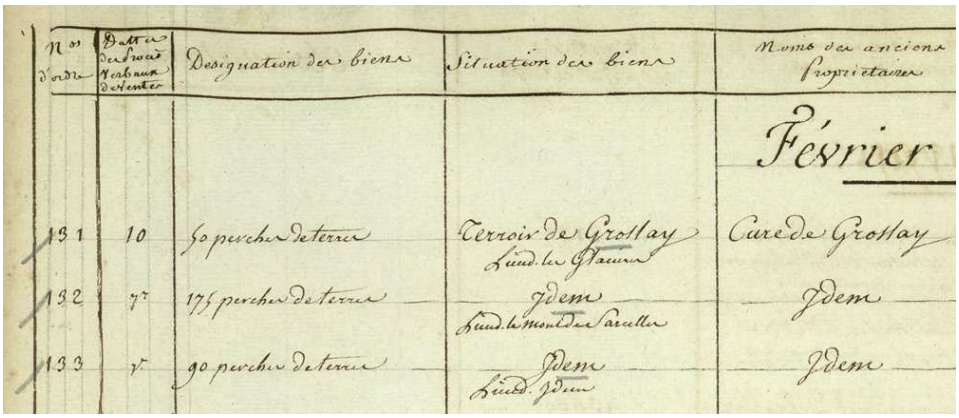


FIGURE 6.5 – Extrait d'un registre de ventes dont on souhaite extraire les numéros de vente (colonne de gauche). *Rappel des figures 6.3 et 6.4.*

Dans ce cas, on va supposer :

1. que le seul problème pouvant se poser est de « manquer » un numéro de vente qui serait effacé ;
2. qu'il n'est pas possible de détecter automatiquement ce problème.

Description avec interaction spontanée Nous allons donc nous baser sur une description simplifiée, présentée ci-après, du contenu de la page. La seule utilisation de l'opérateur

spontaneous concerne l'appel à *reconnait_no*.

$$
\begin{aligned}
&\mathit{colonne\_no} \longrightarrow \mathit{var}\ \ \mathrm{posCol} \\
&\qquad @(\mathit{gauchePage}) \\
&\qquad (\mathrm{posCol}, \_) \leftarrow \mathit{localiser\_colonne\_no} \\
&\qquad @(\mathrm{posCol}) \\
&\qquad \mathit{spontaneous}(\texttt{no\_vente}, \mathit{localiser\_lire\_no}) \\
&\mathit{localiser\_lire\_no} \longrightarrow \mathit{var}\ \ \mathrm{posNo}\ \ \mathrm{posPreciseNo}\ \ \mathrm{ResReco} \\
&\qquad (\mathrm{posNo}, \_) \leftarrow \mathit{localiser\_no} \\
&\qquad @(\mathrm{posNo}) \\
&\qquad (\mathrm{posPreciseNo}, \mathrm{ResReco}) \leftarrow \mathit{get\_answer\_or\_try}(\texttt{no\_vente}, \\
&\qquad\qquad \mathit{reconnait\_no}) \\
&\qquad @(\mathit{sous}\ \ \mathrm{posNo}) \\
&\qquad \mathit{localiser\_lire\_no} \\
&\mathit{localiser\_lire\_no} \longrightarrow \mathit{passer}
\end{aligned}
\tag{6.53}
$$

Stratégie globale De même que précédemment, il est nécessaire de définir une stratégie globale minimaliste pour pouvoir imaginer le comportement du module d'interprétation de page et visualiser l'évolution du contenu de la mémoire associée à une page donnée. L'algorithme 3 présente une stratégie adaptée à une interaction *spontanée* où chaque page est traitée séparément, en étant d'abord interprétée par le module d'interprétation de page, puis contrôlée par un opérateur humain qui peut proposer des corrections et déclencher une nouvelle passe le cas échéant. Une page sera considérée comme complètement traitée si l'opérateur humain ne fait aucune modification dans la mémoire visuelle associée.

Algorithm 3 Exemple de stratégie globale pour l'interaction spontanée

```
1:  procedure TRAITER_LOT(lot)
2:      while lot ≠ ∅ do
3:          p ← lot.extraire()
4:          p.cpt ← p.cpt + 1                ▷ p.cpt compte le nombre de passes sur p
5:          m ← BDC.extraire_memoire(p)
6:          (r, Δm) ← MIP.interpreter(p, m)
7:          m ← fusionner(m, Δm)                                   ▷ point (A)
8:          Δm ← IHM.controler(p, m)
9:          if Δm ≠ ∅ then
10:             m ← fusionner(m, Δm)                               ▷ point (B)
11:             lot.ajouter(p)                                     ▷ Il faut réinterpréter p
12:         end if
13:         DBC.stocker_memoire(m)
14:     end while
15: end procedure
```

Évolution du contenu de la mémoire visuelle associée à l'image En supposant que le second numéro de vente est effacé, voici l'évolution de la mémoire associée à la page de la figure 6.5. Nous nous intéressons, comme précédemment, au contenu de la mémoire visuelle *m* associée à la page *p* après l'exécution des instructions aux points **A** (après la réponse du module d'interprétation de page) et **B** (après la réponse d'un opérateur humain) dans l'algorithme 3.

Contenu de la mémoire visuelle *Itération 1, point* (A)

{ (zoneColonne, `question_optionnelle`, { `type_attendu: no_vente` }),
(zoneNo1, `no_vente`, { `valeur: 131` }),
(zoneNo3, `no_vente`, { `valeur: 133` }) }

Observations : Le module d'interprétation de page n'a localisé que deux nombres sur les trois à extraire, mais une question optionnelle autorise l'ajout (et éventuellement la modification) d'éléments.

Contenu de la mémoire visuelle *Itération 1, point* (B)

{ (zoneColonne, `question_optionnelle`, { `type_attendu: no_vente` }),
(zoneNo1, `no_vente`, { `valeur: 131` }),
(zoneNo2Precise, `no_vente`, { `valeur: 132` }),
(zoneNo3, `no_vente`, { `valeur: 133` }) }

Observations : Un opérateur humain a localisé et donné la valeur du nombre manquant. Il est inutile de supprimer la question optionnelle.

Ensuite, lors de la seconde itération, les données sont réintégrées par le module d'interprétation de page, validées par l'opérateur humain, et la page est sortie du lot à traiter.

Pour conclure, notons que l'opérateur proposé permet d'identifier l'information à propos de l'image considérée qu'il est possible d'apporter spontanément au module d'interprétation de page pour faire progresser son interprétation. Apporter d'autres informations n'aurait, à priori, pas d'impact et il faudrait passer à une édition manuelle pour dépasser les limites de la description pour corriger d'autres problèmes.

6.3.4 Synthèse des opérateurs proposés

Cette section 6.3 a présenté rapidement comment construire automatiquement, grâce la compilation d'une description, un programme d'interprétation symbolique du contenu d'une page (dès lors qu'on dispose de détecteurs d'objets visuels primitifs), et a décrit en détail comment rendre interactif un tel programme.

L'approche que nous avons proposée est basée sur la décomposition d'un mécanisme de détection et correction d'erreur pour permettre son fonctionnent asychrone, dans des processus distincts. Les étapes de résolution d'un problème sont alors :

La détection d'erreur La détection automatique de problème peut être basée sur le constat d'une incohérence entre plusieurs données extraites, sur l'impossibilité de détecter un élément indispensable, sur la détection d'une construction problématique qu'on aurait décrite, ou encore sur la détection d'un cas non prévu.
Lorsque c'est possible, le concepteur du module d'interprétation de page exprime ces

connaissances dans la description de la page, ce qui permet la mise en œuvre d'un mécanisme de détection automatique, spécifique au type de contenu interprété, qui générera des questions (avec l'opérateur *raise_question*) réclamant la connaissance manquante à l'environnement du module d'interprétation de page.
Si la détection automatique d'erreur n'est pas possible, le mécanisme d'interaction *spontanée* autorise une détection (plus coûteuse à l'exécution, mais moins à la conception) hors du module d'interprétation de page, grâce à l'opérateur *spontaneous*.

La correction d'erreur Elle est toujours faite en fournissant au module d'interprétation de page l'information qui lui manque, c'est à dire qui n'est pas dans l'image courante ou dans la connaissance à priori qu'il utilise. Cette correction impose de réviser partiellement l'interprétation proposée par le module d'interprétation de page à l'itération précédente. La réinterprétation de la page, et l'annotation des fragments de la description qui pourront être ignorés avec l'opérateur *raise_question*, autorisent la mise en place d'une réintégration automatique de cette information pour faire progresser l'interprétation, sans perturber la façon de décrire le contenu d'une page.

La récupération sur erreur Cette étape optionnelle permet de faire progresser l'interprétation d'une page dans des parties indépendantes lorsqu'un problème est détecté, afin de générer plusieurs questions à chaque itération et accélérer le processus de correction. L'opérateur *catch_question* permet au concepteur de structurer la description de la page en blocs qui peuvent être interprétés indépendamment.

Ces étapes essentielles sont gérées automatiquement grâce aux propriétés exprimées par les opérateurs que nous avons proposés. L'implémentation de ces derniers peut évoluer sans nécessiter de modification dans la description d'une page. Il est également possible de se servir du premier niveau de description pour ajouter facilement de nouveaux éléments, comme par exemple pour indiquer qu'un élément est indispensable dans la page :

$$
\begin{array}{l}
\mathit{indispensable}(T, R_{\text{auto}}) \longrightarrow \\
\quad \mathit{var}\ \ z_R \\
\quad z_R \leftarrow \mathit{acces\_zone\_recherche} \\
\quad \mathit{catch\_question}(T, \\
\qquad R_{\text{auto}} \mid \mathit{raise\_question}(z_R, T, \{\ \texttt{txt}: \text{"Elément requis manquant [...]"}\ \}))
\end{array}
\tag{6.54}
$$

6.4 Conclusion

Dans ce chapitre, nous avons proposé des outils et des méthodes qui permettent de concevoir simplement un module d'interprétation de page qui s'intègre dans un mécanisme global et itératif d'interprétation, et échange de façon asynchrone des connaissances avec son environnement. Ces éléments sont facilement réutilisables, et ce à trois niveaux différents et relativement indépendants entre eux.

1. Les éléments de la section 6.1 visent à donner des outils de niveau algorithmique permettant la mise en place d'une communication entre le module d'interprétation de page et son environnement. Elle décrit globalement la gestion de la mémoire visuelle qu'il est nécessaire de mettre en place pour pouvoir modifier le comportement du module en lui apportant de nouvelles données.
   Ces éléments peuvent être repris par toute approche *algorithmique*, pour reprendre la terminologie du chapitre 4.

2. Les éléments de la section 6.2 constituent une formalisation rigoureuse d'un module d'interprétation de page minimaliste mais réaliste (car capable de retour arrière) généré automatiquement par la compilation d'une description du contenu de la page. La formalisation de ce socle nous permet alors de spécifier précisément la gestion de la mémoire à mettre en œuvre.
   Cette formalisation peut être utile à toute approche basée sur une description déclarative, et générant une analyse guidée par le but.
3. Les éléments de la section 6.3 donnent un ensemble de propriétés utiles pour gérer automatiquement l'échange d'information entre le module d'interprétation de page et son environnement afin de faire progresser l'interprétation, grâce à l'expression de propriétés naturelles à propos du contenu des pages.
   L'identification de ces propriétés pourra intéresser tout concepteur de système d'interprétation de documents, et les règles de compilation proposées pour les opérateurs permettent l'exploitation de ces mécanismes presque directement pour les approches basées sur une analyse guidée par le but.
   Nous pensons en effet que les opérateurs proposés peuvent être adaptés à des approches guidées par les données (dites« ascendantes ») à condition de mettre en œuvre une fonction *conserver_resultat*. La communauté de la compilation de langages dispose d'une littérature importante sur le sujet, ainsi que de représentations appropriées pour décrire les règles de compilation des opérateurs : construction monadiques, etc.

La conception d'un module d'interprétation de page à l'aide des outils proposés autorise la mise en place rapide de scénarios d'interprétation qui permettent de tirer profit d'informations contextuelles, ce qui facilite alors la production de résultats fiables, et limite l'effort de correction nécessaire, comme nous le verrons dans le chapitre 8. Dans la mesure où l'information contextuelle fournie concerne le contenu de l'image, et non des paramètres du système comme un seuil de binarisation par exemple, les opérateurs humains peuvent directement participer à l'apport d'information, de façon homogène aux processus automatiques, et surtout de façon complémentaire, car ils excellent particulièrement dans la production de résultats abstraits et riches en contexte.

Troisième partie

— Validation — Réalisation et exploitation d'un système complet

Chapitre 7

Réalisation d'un système complet

Introduction

Dans la partie précédente de ce document, nous avons spécifié, dans le chapitre 5, l'architecture globale d'un système permettant l'interprétation *contextuelle* et *assistée* de fonds documentaire, basée sur un mécanisme d'interprétation *itératif* autorisant l'échange *asynchrone* d'informations entre les parties du système. Dans le chapitre 6, nous avons ensuite montré comment il est possible de générer le programme d'un module d'interprétation de page, qui gère automatiquement l'échange d'informations avec son environnement, grâce à une extension simple et peu perturbante du langage de description des pages. Le système générique que nous avons proposé permet une diminution potentielle de l'effort de correction manuel requis en post-traitement, pour des scénarios bien construits, sans augmenter la difficulté de la conception nécessaire à la spécialisation du système à un fonds documentaire donné.

Dans ce chapitre, nous allons montrer que nos propositions peuvent être appliquées pour réaliser un système complet, intégrant un certain nombre de modules de diverses origines, et que l'extension d'un module d'interprétation de page peut être effectivement réalisée selon nos recommandations.

Du point de vue des concepteurs et développeurs, notre travail a donc permis de combiner facilement les outils réalisés par différentes personnes sans perturber la description du contenu d'une page qui reste unitaire, ni le traitement page par page habituel. Le découpage automatique de l'interprétation de chaque page en plusieurs tâches, traitées par des modules divers communiquant de façon asynchrone, est déterminé par la connaissance disponible à propos des pages et du fonds, ainsi que par l'état des données à traiter.

Ce chapitre reprend la progression de la partie II pour permettre une description globale de la mise en œuvre du système, avant de se focaliser sur l'extension d'un module d'interprétation de page existant à l'aide de nos propositions.

- La première section dressera donc un tableau large des modules mis en œuvre, les technologies utilisées, et indique les principaux auteurs afin de positionner plus clairement nos travaux. Elle permettra de donner une présentation générale des composants utilisés dans les expérimentations qui seront présentées au prochain chapitre.
- La seconde section présentera la méthode DMOS-P, qui permet la génération automatique de modules d'interprétation de page, et comment nous avons complété cette méthode pour permettre la génération d'un nouveau type de module adapté à un mécanisme d'interprétation global itératif.

7.1 Vision d'ensemble du système

Le système développé a respecté trois principes fondateurs.

1. Faciliter l'enchaînement de traitements simples, faciles à concevoir et à réaliser, qui peuvent être soit automatiques, soit impliquer des opérateurs humains. Certains traitements peuvent travailler sur des données unitaires (page, zone de page) mais beaucoup nécessitent de considérer un ensemble de données, en particulier si des opérateurs humains sont sollicités, et travaillent de façon asynchrone.
2. Permettre l'indexation et le regroupement des données produites dans différentes pages, pour autoriser des mécanismes de validation, de correction, d'apprentissage ou simplement une utilisation finale (recherche, principalement).
3. Autoriser l'interprétation en plusieurs passes de chaque image en limitant le surcoût en complexité de conception et en temps de calcul.

Intégrant dans notre démarche la contrainte liée à la quantité des données à traiter, nous nous sommes assez naturellement tournés vers des outils et méthodes dédiés à l'analyse massive de données, tels ceux utilisés par les moteurs de recherche récents. En particulier, nous nous sommes intéressés à des approches telles que « *map-reduce* » [31] permettant la répartition de calculs sur une grille de machines [1] en minimisant les transferts de données [2] ; ou à des bases de données autorisant une grande flexibilité dans le schéma utilisé [16].

Au niveau de l'architecture globale du système, le *framework* Apache Hadoop [3] dédié à l'analyse distribuée de données (et utilisé par des sociétés telles que *Yahoo!* et *Amazon*) a été un des supports du projet.

Si notre utilisation de ces outils extrêmement performants n'a sans doute été que superficielle dans la mise en œuvre de ce système expérimental, pour des raisons de temps ou d'infrastructures disponibles, nous avons toutefois pris soin de conserver, dans notre démarche, la possibilité d'un passage à l'échelle en respectant le mode de conception associé à ces approches.

7.1.1 Base de donnée centrale

La base de donnée centrale, au cœur du système, a été très rapidement mise en œuvre à l'aide de l'outil Apache HBase [4], partie intégrante du projet Apache Hadoop.

Cet outil nous a permis :

– d'étendre progressivement l'ensemble des types de données gérés par le système ;
– de réaliser simplement des requêtes collectant les données associées à une page pour constituer la mémoire visuelle nécessaire à l'interprétation de cette dernière ;
– de réaliser simplement des requêtes collectant des données d'un certain type (patronymes par exemple) sur un ensemble de pages pour procéder à des regroupements.

Deux dispositifs sont nécessaires dans ce système pour permettre le stockage de deux types de données : (i) les données statiques, principalement les images à interpréter ; (ii) les données dynamiques, c'est à dire celles produites et modifiées par le système. Apache HBase s'est avéré un excellent choix pour le second cas.

1. On utilisera dans la suite le terme « grille » pour parler d'un ensemble de machines dédiées au calcul, et le terme « cluster » pour parler d'un regroupement d'objets à classer, tels que des vignettes de mots.
2. Dans le cas d'un mécanisme global itératif, la possibilité de ne pas avoir à transférer à nouveau chaque image vers la machine de calcul est un avantage certain.
3. `http://hadoop.apache.org/`
4. `http://hbase.apache.org/`

Parmi les données stockées, on peut distinguer, d'une part, celles rattachées à une position précise dans une image, comme les patronymes, noms de communes ou nombres détectés. Ces données possèdent généralement les informations suivantes :

- une zone indiquant la surface concernée dans l'image ;
- une liste d'hypothèses produites par un ou plusieurs classifieurs ;
- une valeur finale validée si nécessaire ;
- une chaîne de caractères indiquant le type de donnée.

D'autre part, certaines données sont rattachées à un niveau physique du fonds documentaire (livre, registre, etc.), et sont partagées par une ou plusieurs pages. C'est par exemple le cas des lexiques de patronymes ou de noms de communes, qui ont pu être mis à jour dès qu'un nouvel élément était validé dans les pages de leur portée, et qui ont pu être appliqués à d'autres parties du fonds au cours de traitements ultérieurs.

L'intégration de cet outil a principalement été réalisé par L. Guichard, avec qui le schéma général de stockage des données a été élaboré.

7.1.2 Module de stratégie globale et pilotage du système

Le module de stratégie globale a été développé selon les besoins spécifiques du projet, sur la base d'un ensemble de librairies facilitant l'invocation d'un enchaînement de traitements sur une grille de calcul. La stratégie globale est définie sous forme d'un automate indiquant pour chaque ensemble de données à traiter les états possibles et les transitions entre ces états. Un moteur de gestion d'événements générique permet de n'avoir à spécifier qu'un nouvel automate pour mettre en œuvre un nouveau système capable d'utiliser les modules de calcul disponibles.

L'absence d'un environnement de calcul entièrement basé sur l'architecture Hadoop au sein de notre laboratoire au moment des premières expériences à justifié le développement de solutions internes à l'équipe qui, tout en restant compatible avec le modèle de distribution des calculs envisagé, nous a laissé toute la flexibilité nécessaire à la réalisation d'un prototype. Ceci a toutefois requis le développement d'un ordonnanceur chargé de répartir des traitements sur la grille de calcul.

Il peut être intéressant de noter que nous n'avons pas cherché à définir un langage de description au niveau du fonds documentaire [5], qui aurait décrit les types de pages à interpréter et les informations à extraire. Bien qu'il soit envisageable d'imaginer un échange d'information entre des groupes de pages aux contenus liés (typiquement pour les documents sur quelques pages, comme des factures ou des décrets), il nous semble plus intéressant d'intégrer ces connaissances directement en base de donnée en associant aux pages leur type, puis en gérant la spécificité de ces données au niveau de la stratégie globale. Nous donnons ci-après quelques raisons en faveur de ce choix.

- Une expression régulière est largement suffisante, en général, pour pour exprimer la séquence des types de pages au sein d'un document. Pour un livre, par exemple, on pourrait avoir un expression du type :

$$\begin{aligned} &\mathit{couverture}\ \ \mathit{page\_blanche}^{+}\ \ \mathit{sommaire}^{*}\ \ \mathit{page\_blanche}^{*} \\ &(\mathit{page\_texte}\ \mid\ \mathit{page\_illustration})^{*}\ \ \ldots \end{aligned}$$

  Cette connaissance est exprimable facilement au niveau de la stratégie sans nécessiter de formalisation particulière.

5. Malgré certains travaux préliminaires dans cette direction [17].

– Il est possible d'apporter simplement et rapidement cette information au niveau des pages en base, en particulier grâce au classement défini par les documentalistes (voir sous-section 1.1.2) qui reste disponible (parfois de façon assez minimaliste, il est vrai) après numérisation des pages. La définition et la gestion de groupes de pages abstraits grâce à l'association facile d'informations à chaque niveau du fonds documentaire est alors un atout.
– Il est possible de détecter automatiquement le type des pages, si c'est nécessaire.
– Finalement, il nous a semblé plus pertinent de nous intéresser à la systématisation de la circulation d'informations selon certains schémas typiques qui semblent se dégager (apprentissage de dimensions, optimisation des hypothèses d'une séquence, etc.) mais nos travaux dans cette direction sont encore à un stade préliminaire.

À titre d'exemple, on peut indiquer que l'utilisation de connaissances à priori sur la structuration du fonds documentaire nous a été particulièrement utile dans le cas des documents d'archives du XVIII^{e} siècle que nous avons traités. Au cours des différents travaux que nous avons menés, nous avons eu besoin d'optimiser les hypothèses de reconnaissance des numéros de ventes contenus dans les tableaux de ventes (les « registres »). Grâce aux informations disponibles après numérisation, il a été facile d'identifier les pages contenant ces numéros, et de délimiter les registres qui contiennent des numérotations séparées. Une exploitation automatique de la propriété de croissance avec un incrément de 1 de ces numéros a pu être exploitée de façon quasi-automatique avec un mécanisme générique (basé sur un algorithme de Viterbi modifié) au niveau du fonds documentaire.

Les outils et librairies permettant la construction d'un module de stratégie globale ont presque intégralement été développées par L. Guichard, lors de la mise au points des scénarios d'interprétation du fonds documentaire du XVIII^{e} siècle sur lequel nous avons travaillé.

7.1.3 Module d'interprétation de page

La réalisation du module d'interprétation de page a été possible grâce à l'environnement de développement DMOS-P. Nous nous intéresserons plus particulièrement à cet environnement et à ce qu'apporte notre approche dans ce contexte dans la prochaine section.

D'un point de vue purement technique, l'intégration des outils existants, permettant à l'origine de générer automatiquement un module d'interprétation de page travaillant sur des pages isolées, a nécessité de mettre en conformité les programmes générés avec l'interface standard des modules de traitement du système.

Les travaux sur lesquelles reposent nos propositions sont ceux de J. Camillerapp, B. Coüasnon, et A. Lemaitre.

7.1.4 Interfaces homme–machine

Il a été nécessaire de réaliser deux interfaces homme–machine pour le projet d'interprétation de registres de ventes du XVIII^{e} siècle. Basées sur un même socle commun, ces interfaces permettent de :

1. visualiser un ensemble de mots similaires (cluster) afin d'éliminer les intrus et valider l'étiquette associée ;
2. visualiser les éléments reconnus et les questions associées à une page pour permettre de corriger spontanément des erreurs et répondre aux questions.

La figure 7.1 montre l'interface utilisée dans le second cas.

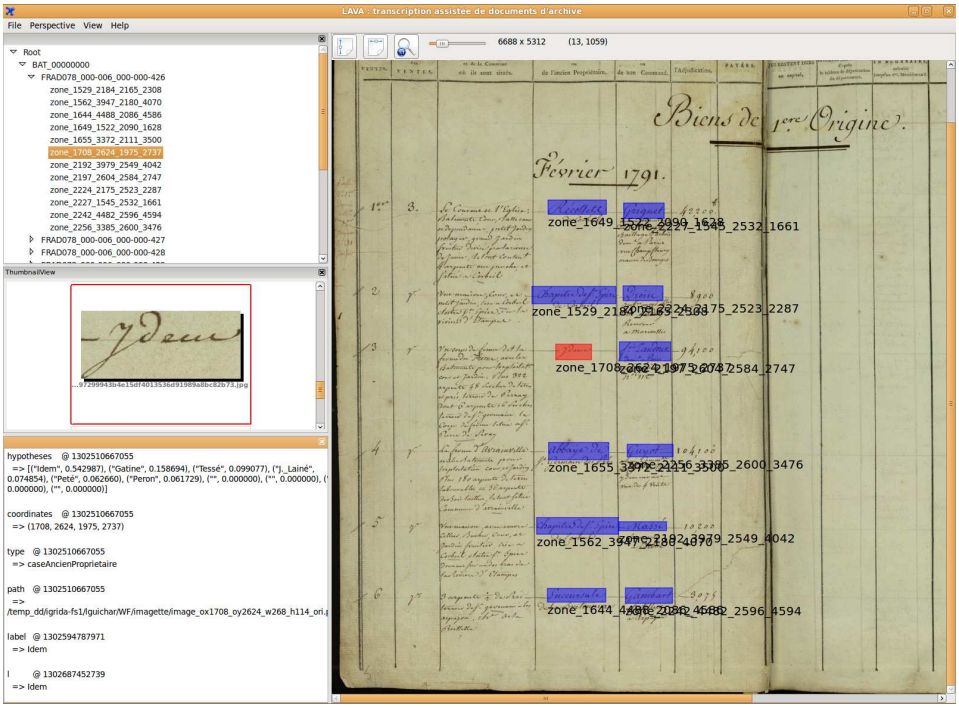


FIGURE 7.1 – Exemple d'interface dédiée à la correction d'erreurs de reconnaissance. Dans cette illustration, différentes vues d'une donnée problématique (pour laquelle une question a été posée) permettent à l'opérateur humain de comprendre la situation et saisir la valeur effective du mot.

Dans le premier cas, chaque interface reçoit du module de stratégie globale une liste de clusters de mots à valider, et renvoie des clusters modifiés : il est possible de supprimer des éléments du cluster, et de modifier l'étiquette globale.

Dans le second cas, l'interface reçoit une ou plusieurs images à interpréter et le contenu de la mémoire associée. L'interface renvoie, après interaction avec un opérateur humain, un ensemble de modifications à apporter à la mémoire visuelle de l'image.

Le module de stratégie globale que nous avons réalisé permet la gestion simultanée de plusieurs interfaces homme–machine, et donc la distribution automatique du travail manuel à un ensemble d'opérateurs humains.

Finalement, au niveau du développement, la gestion des règles de présentation et d'édition des différentes données extraites a été intégrée dans les librairies servant de base à la construction de nouvelles interfaces. Afin de faciliter la création de ces règles, la définition des types de données (colonne de numéros d'ordre, patronymes, lexiques, nombres, noms de communes, etc.) a été centralisée, dans la mesure du possible, au niveau global du système.

Ces outils ont été développés par L. Guichard.

7.1.5 Autres modules

Comme nous l'avions mentionné au chapitre 5, d'autres modules sont nécessaires en pratique pour construire des scénarios applicables à des cas réels. En voici deux que nous

avons utilisés dans le cadre de ce projet.

7.1.5.1 Reconnaisseurs de nombres et de mots manuscrits

Les travaux de B. Coüasnon, L. Guichard et A. Toselli [45, 46, 44] ont permis la réalisation d'outils de reconnaissance de texte manuscrit (nombres et mots) performants. Il est possible de fournir à ces outils un certain nombre de contraintes quant aux valeurs acceptables par le reconnaisseur. Dans le cas des nombres, il s'agit de décrire les valeurs minimale et maximale possibles. Dans celui des mots, il est possible de fournir un modèle de langage adapté au problème considéré : un dictionnaire peut être utilisé si les mots sont connus à l'avance, et si ce n'est pas le cas, un modèle de co-occurrence de lettres de type n-grammes est utilisable en solution de repli.

Ces outils fonctionnent selon une analyse en deux temps, générant des hypothèses de segmentation à l'aide d'un modèle de Markov caché (HMM), et réévaluant le score des hypothèses à l'aide d'un séparateur à vaste marge (SVM).

La difficulté de la lecture automatique de contenus manuscrits est alors surmontée grâce à l'exploitation du contexte documentaire matérialisé par une circulation d'information selon deux directions.

1. De la page vers le mot : la connaissance du type de contenu attendu permet de contraindre le domaine de valeurs qui peuvent être produites par le reconnaisseur. Dans le cas des registres de ventes, la description de la page permet de savoir, en fonction de la colonne du tableau dans laquelle se trouve la zone à reconnaître, s'il faut demander la reconnaissance d'un nombre, d'un nom de commune appartenant à un lexique connu, ou d'un patronyme pour lequel on ne dispose pas de lexique fiable.
2. Du mot vers la page, voire vers le fonds documentaire : grâce à la génération de liste d'hypothèses, il est possible de retarder la prise de décision tant que le système ne dispose pas d'assez d'informations pour procéder à une optimisation, si c'est possible.

Dans le cas des registres de ventes du XVIII^{e} siècle, voici un exemple de mécanisme d'optimisation que nous avons utilisé. Dans le contexte de la reconnaissance de patronymes des anciens propriétaires ou des acheteurs, il est très probable que deux patronymes visuellement similaires présents dans des pages voisines fassent référence à la même personne. Il est donc possible de regrouper ces éléments pour croiser les hypothèses d'interprétation avec des méthodes de *clustering*, et de fusion de classifieurs.

Ces outils de reconnaissance ont pu être intégrés au sein du module d'interprétation de page, ou dans des modules séparés, selon les besoins, sans nécessiter de modification de la description de la page, à part lors de l'appel au reconnaisseur : si le reconnaisseur qui doit être utilisé est implémenté dans un autre module, alors une question est générée et le module de stratégie globale se chargera de transmettre cette demande et de faire circuler l'information nécessaire. La fragmentation de l'interprétation de la page entre plusieurs modules est gérée automatiquement grâce au mécanisme d'interprétation itératif.

Des outils OCR pourraient être intégrés de la même façon, et si nous n'avons pas eu besoin de reconnaître de caractères imprimés dans le cas des registres de ventes, il est intéressant de noter que le mécanisme d'interprétation itératif permet de disposer d'une machine dédiée à l'analyse OCR vers laquelle sont routées toutes les questions relatives au contenu de zones identifiées par le module d'interprétation de page. Ceci permet de gérer d'éventuels problèmes de licence qui peuvent gêner le déploiement de librairies OCR sur toute une

grille de calcul, comme dans le cas des dernières versions de *FineReader* que nous avons eu l'occasion de tester.

7.1.5.2 Regroupement de mots visuellement similaires

Grâce aux travaux de J. Camillerapp et L. Guichard, il a été possible de regrouper des mots visuellement similaires pour permettre une fusion des hypothèses de reconnaissance pour chacun des éléments composant les clusters ainsi formés. Ce module de *clustering* fonctionne selon la procédure suivante :

1. les zones des mots (« vignettes » ci-après) intéressants (des patronymes par exemple) sont localisées lors de l'interprétation de la page par le module d'interprétation de page ;
2. ensuite, pour des pages voisines (une dizaine de pages par exemple), les vignettes sont comparées deux à deux à l'aide d'une méthode à base de points d'intérêt pour construire une matrice d'indices de similarité ;
3. un algorithme de *clustering* agglomératif hiérarchique regroupe ensuite les éléments les plus semblables, jusqu'à un seuil de ressemblance minimale ;
4. les listes d'hypothèses à l'intérieur d'un cluster sont fusionnées pour produire une nouvelle liste d'hypothèses.

Nous avons identifié deux modes de fonctionnement extrêmes possibles au niveau de ce module :

- un fonctionnement n'autorisant que la production de réponses très fiables, dans un but de réapprentissage par exemple : dans ce cas, le *clustering* utilisera une fonction de distance peu tolérante (comme le maximum des distances à chacun des éléments du cluster) pour intégrer un nouvel élément, et la fonction de fusion des listes d'hypothèses ne conservera que les hypothèses présentes pour chaque membre du groupe ;
- d'un autre côté, un objectif d'indexation visera à proposer des listes d'hypothèses plus complètes et moins fiables, favorisant les regroupements (avec une distance à un cluster basée sur un minimum, par exemple) et autorisant certaines hypothèses isolées à survivre dans la liste finale.

Ces modes seront illustrés dans une des expériences du prochain chapitre.

Cette première section montre une certaine flexibilité au niveau des modules qui peuvent être intégrés. Ce système permet de fédérer les travaux de diverses origines et la construction de flots de traitements riches. Les modules intégrés peuvent travailler page par page, sur un ensemble de pages, ou sur des ensembles de données plus abstraits, comme des vignettes de patronymes manuscrits par exemple. L'interaction asynchrone entre les modules autorise ces derniers à avoir recours, si nécessaire, à des opérateurs humains. Finalement, le mécanisme d'interprétation itératif permet la réintégration automatique au sein du module d'interprétation de page des données produites dans d'autres modules.

7.2 DMOS-PI : une extension de l'approche DMOS-P

Cette seconde section présente ce que l'intégration de nos propositions au système de construction de modules d'interprétation de page DMOS-P a apporté à ce dernier. Nous commençons par présenter l'écosystème de cette approche, puis nous détaillons en quoi

l'intégration d'un mécanisme d'interprétation itératif a permis de renforcer la gestion de l'incertain dans les modules d'interprétation générés automatiquement.

7.2.1 À propos de l'approche DMOS-P

DMOS-P, que nous avons déjà présenté superficiellement dans la sous-section 2.3.3, est un ensemble d'outils, dont un *framework* et des librairies de base, un compilateurs, et des extracteurs de primitives visuelles, réalisés autour d'une méthode d'interprétation d'image de documents. Un des principes fondateurs de cette approche est qu'il faut être capable de générer automatiquement un programme d'interprétation dédié à un type de document précis à partir d'une description spécifique des contenus à extraire, reconnaître et structurer.

L'ensemble des outils génériques qui peuvent être utilisés pour fabriquer un module d'interprétation spécifique et permettent son exécution forment un écosystème dont on présente ici les principaux éléments.

Langage de description La description du contenu de la page est réalisée à l'aide d'un langage grammatical déclaratif appelé EPF (pour *Enhanced Position Formalism*). Facilement extensible, il permet la génération d'un programme capable de prédire la position des éléments à interpréter, et doté de mécanismes facilitant la gestion des éléments parasites dans le signal de l'image, comme nous le verrons en sous-section 7.2.1.2.

Compilateur Un compilateur permet de générer automatiquement, à partir d'une description spécifique des contenus d'un type de document, un programme capable d'interpréter une image et de produire une représentation informatique des contenus.

Moteur d'interprétation Un ensemble de librairies et un moteur d'interprétation sont à la base du programme généré, et en forment la partie générique. Un module d'interprétation de page construit avec la méthode DMOS-P gère automatiquement une recherche guidée par le but, en profondeur d'abord (par défaut) d'un ensemble de contenus conformes à la description fournie. Une représentation avec plusieurs niveaux de résolution permet une meilleure extraction des informations contenues dans l'image.

Extracteur de primitives Afin de permettre au moteur d'interprétation (symbolique) de l'image de fonctionner, il est nécessaire de détecter dans l'image des éléments visuels primitifs qui deviennent les *terminaux* de la description grammaticale utilisée. Ces terminaux peuvent être, en pratique, des segments (à tendance horizontale, verticale, ou autre), des composantes connexes, des lignes…

L'ensemble de ces éléments ressemble fortement, à un niveau macroscopique, au module d'interprétation de page à titre indicatif en sous-section 6.2.2. Cependant, il faut préciser que DMOS-P permet la génération de programme d'interprétation utilisés en production pour le traitement de fonds de plusieurs dizaines de milliers de pages. La méthode de positionnement, les techniques de gestion du bruit et de l'incertitude en général font de ce système un objet beaucoup plus complexe que celui qui nous a servi de référence, même si nos propositions ont bien pu être intégrées avec la méthode décrite dans le chapitre 6.

Ce qui suit instancie quelques-uns des éléments que nous venons de présenter.

7.2.1.1 Moteur d'interprétation

Le moteur d'interprétation est au cœur des programmes construits avec la méthode DMOS-P. Il est basé sur un mécanisme d'interprétation de l'image à partir d'une descrip-

tion grammaticale conçu à l'aide d'une variante de Prolog, à savoir une implémentation λProlog [102, 13]. Ce langage propose des constructions d'ordre supérieur, permettant l'intégration de contexte dans la preuve de buts et la contruction rapide de nouveaux éléments de langages [6]. À l'instar des DCG *(Definite Clause Grammars)*, les grammaires logiques proposées par les implémentations Prolog classiques, PROLOG/MALI (l'implémentation de λProlog utilisée) dispose d'une variante (plus puissante) appelée λHHG, pour *Higher-order Hereditary Harrop Grammar* (grammaire en formules héréditaires de Harrop d'ordre supérieur) [51].

7.2.1.2 Langage de description

Afin d'intégrer les constructions spécifiques à l'organisation bidimensionnelle des objets graphiques des images de documents, DMOS-P propose le langage EPF qui sert de base à la spécification du problème, en décrivant en particulier :

- l'organisation des contenus d'un document ;
- les contraintes sémantiques entre ces derniers ;
- les propriétés utiles permettant la gestion automatique du bruit et des ambiguïtés ;
- la construction de la structure résultant de l'interprétation de l'image.

Ce langage utilise intensivement les constructions d'ordre supérieur pour proposer des *opérateurs* de description dont voici les principaux.

Opérateur de positionnement L'opérateur de positionnement permet de modifier la zone de recherche du prochain symbole (terminal ou non) à reconnaître. Il s'appuie sur une relation spatiale qui peut être spécifiée par l'expert écrivant la description. Il permet également d'ordonner les éléments à l'intérieur de la zone de recherche pour identifier de façon unique le prochain terminal à considérer. Exemple :

```
bordureGauche BG && AT(enBas BG) && bordureBas BB
```

où `&&` est l'opérateur de concaténation.

Opérateur d'accès aux terminaux Le système propose par défaut deux types de terminaux : les segments (objets linéaires) et les composantes connexes (objets bidimensionnels). Il est facile d'intégrer d'autres terminaux, comme des lignes de texte par exemple. Ces opérateurs sont les suivants :

```
TERM_SEG <PreCond> <PostCond> <Label> <Seg>
TERM_CMP <PreCond> <PostCond> <Label> <Comp>
```

où `<PreCond>` (resp. `<PostCond>`) sont des conditions (géométriques ou sémantiques) testées avant (resp. après) sélection d'un terminal (permettant un filtrage du bruit), et `<Label>` est un marqueur utilisé pour construire l'objet manipulé (`<Seg>` ou `<Comp>`).

Opérateur de recherche L'opérateur de recherche permet la gestion du bruit et des ambiguïtés au niveau des non-terminaux. Il indique la nécessité d'un élément. Au niveau de l'implémentation, ceci a pour effet de modifier ponctuellement la stratégie de recherche de solution : normalement en profondeur d'abord *(depth-first search)*, celle-ci devient ponctuellement en largeur d'abord *(breadth-first search)* pour forcer la détection du symbole, en général structurant pour le reste de l'analyse.

6. Cette présentation est, bien entendu, extrêmement réductrice.

```
FIND(Regle) UNTIL(ConditionArret)
```

Contraintes sémantiques Pour intégrer des contraintes sémantiques dans la description et permettre leur validation, il est possible d'ajouter des fragments de programmes Prolog dans les descriptions, travaillant sur les variables locales à une règle de description. Ces contraintes sont exprimées de la façon suivante :

```
{   code Prolog   }
```

7.2.1.3 Exemple de description

Afin de mieux présenter le langage utilisé pour la description des documents dans les expériences du prochain chapitre, nous donnons ici un exemple de description écrite dans le langage EPF.

La description suivante est un exemple simple de règle permettant la détection d'un rectangle et le calcul de son aire. Les règles de description sont indiquées en minuscule, et les variables commencent par une majuscule. Les variables se comportent comme dans un programme Prolog habituel : elles sont instanciées grâce à un mécanisme d'unification.

```
rectangle Aire ::=
     TERM_SEG vertical noCond "gauche" Seg1 --> bord_gauche &&
     AT(haut Seg1) &&
     TERM_SEG horizontal noCond "haut" Seg2 &&
     AT(droite Seg2) &&
     TERM_SEG vertical noCond "droit" Seg3 &&
     AT(bas Seg3) &&
     TERM_SEG horizontal noCond "bas" Seg4 &&
     AT(gauche Seg4) &&
     TERM_SEG vertical noCond _ _ <-- bord_gauche &&
     {  calculer_aire Seg1 Seg2 Seg3 Seg4 Aire  }
```

Dans la règle de description précédente, `vertical` et `horizontal` sont des contraintes sur l'orientation des segments, et `noCond` indique une condition toujours vérifiée (absence de contrainte). On voit aussi l'utilisation d'opérateurs d'affectation (`-->`) et de lecture (`<--`) servant à vérifier que le bord gauche est correctement placé par rapport au dernier côté détecté. Le caractère spécial « _ » indique un élément non instancié.

L'appel à cette règle `rectangle` peut être fait de la façon suivante :

```
AT(Position) && FIND(rectangle Aire) UNTIL (noCond)
```

Les modules d'interprétation générés avec DMOS-P permettent donc un positionnement élaboré, une gestion du bruit, la recherche d'éléments structurants, et surtout bénéficient d'un socle extensible permettant l'intégration de nouveaux mécanismes tels que ceux mentionnées en sous-section 2.3.3.

7.2.2 DMOS-PI : une interprétation plus robuste

Nos propositions ont fait l'objet d'une extension de DMOS-P, que nous avons nommée DMOS-PI pour indiquer la capacité des modules générés à s'intégrer dans un mécanisme global *itératif*.

7.2.2.1 Extensions réalisées

L'intégration de nos propositions visant à étendre le langage de description des pages a abouti à la réalisation de deux nouvelles librairies à partir desquelles de nouveaux programmes peuvent être construits.

- La première regroupe les opérations relatives à la gestion de la mémoire visuelle, et permet la construction de nouveaux outils à partir de cette structure. Elle définit les opérateurs *lire_mem*, *consommer_mem*, *ajouter_mem* et *supprimer_mem* présentés en sous-section 6.1.2.3.
- La seconde permet la gestion des mécanismes d'interaction asynchrone basés sur la mémoire visuelle. Elle contient la définition et la mise en œuvre des opérateurs *get_answer_or_try*, *catch_question*, *raise_question* et *spontaneous* présentés en section 6.3.

Cette extension a apporté deux améliorations notables dans la méthode DMOS-P : elle a permis l'exploitation de nouvelles connaissances lors de l'interprétation ; et elle a renforcé le panel de méthodes visant à gérer l'incertain.

7.2.2.2 Connaissances complémentaires

La méthode DMOS-P permettait, initialement, l'exploitation de deux types de connaissances pour interpréter des images de documents [25] :

1. les connaissances à priori, exprimées à l'aide du langage EPF par le concepteur du système, et qui constituent une description des contenus à extraire ;
2. les connaissances extraites de l'image, grâce aux extracteurs de primitives et au mécanisme de multi-résolution.

L'extension que nous avons proposée permet, au prix d'une perturbation minime des descriptions, de tirer profit, lors de l'interprétation d'une image, de connaissances contextuelles qui ne sont pas contenues dans la page. Ces connaissances peuvent alors provenir du fonds documentaire, ou être fournies par un opérateur humain.

7.2.2.3 Gestion des incertitudes

Au niveau de la gestion des incertitudes, plusieurs outils existaient déjà dans la méthode de base pour gérer les problèmes suivants.

Le bruit La présence d'éléments parasites dans l'image, comme par exemple des petits segments ou composantes connexes à cause des dégradations du papier, peut généralement être filtrée grâce aux préconditions passées aux opérateurs `TERM_SEG` et `TERM_CMP` présentés précédemment (sous-section 7.2.1.2).

L'ambiguïté La multiplicité des éléments susceptibles de composer une forme à un moment donné provoque parfois une ambiguïté qui ne peut généralement être levée qu'après la détection d'autres éléments structurants. Par exemple, dans le cas de la description précédente d'un rectangle, si plusieurs segments, à l'endroit considéré, peuvent former un bord gauche, mais que certains sont trop petits, alors il faut être capable de remettre en cause le choix de chaque élément tant que la forme n'est pas détectée complètement. L'opérateur `FIND ... UNTIL` permet de gérer ces cas.

Grâce à l'intégration des opérateurs *get_answer_or_try* et *catch_question*, notre extension permet à présent de gérer les erreurs d'*incohérence*, de *rejet* ou d'*absence* d'éléments

indispensables en définissant un *point de reprise* de l'analyse qui permette tout de même de progresser, au lieu d'être obligé, comme avant, de se contenter d'accepter les résultats locaux les moins improbables. Cette nouvelle gestion reproduit dans une certaine mesure le concept de « mode panique » des compilateurs classiques [3] pour lequel il faut définir, pour chaque type de problème géré :

- les transformations à apporter dans la pile d'analyse pour retrouver un état stable : dans notre cas, il s'agit de déterminer le non-terminal (la règle de description) à partir duquel reprendre l'interprétation ;
- l'élément suivant à trouver dans le flot d'entrée : dans notre cas, l'opérateur de position associée au prochain élément permet normalement de repositionner correctement la zone de recherche.

7.3 Conclusion

Malgré la complexité de l'approche DMOS-P sur laquelle nous avons basé la réalisation d'un prototype, nos propositions ont pu être intégrées facilement et ont donné naissance à une extension de l'approche qui lui permet de tirer profit d'une nouvelle forme de connaissance contextuelle, tout en améliorant sa gestion des problèmes présents dans les documents dégradés ou anciens. Ceci confirme la possibilité d'utiliser nos propositions pour transformer un système existant.

La réalisation d'un prototype de système global mettant en œuvre un mécanisme d'interprétation itératif a permis l'intégration de travaux variés au sein de notre équipe, sans nécessiter de perturbations lourdes dans notre façon de décrire et traiter les pages. On peut noter la possibilité de répartir automatiquement le traitement d'une image entre plusieurs modules, selon la connaissance disponible à propos de cette dernière, dès qu'on enrichit légèrement la description de page pour utiliser les nouveaux opérateurs proposés.

Il a alors été possible de créer plusieurs scénarios dédiés à des types de documents particuliers, tirant profit des modules présentées dans la section 7.1. Le chapitre suivant va montrer, pour quelques scénarios précis (qui autorisent des mesures ciblées), le gain au niveau de l'effort de correction nécessaire pour traiter certains documents.

Chapitre 8

Expérimentations

Introduction

Les chapitres précédents ont montré que nos propositions, c'est à dire les outils que nous avons conçus pour interpréter des fonds documentaires incertains, sont construits sur une base théorique rigoureuse (chapitre 6) dont la réalisation a été validée (chapitre 7), et qu'ils peuvent alors être transposés facilement dans un autre environnement. Ce chapitre, vise à présent, à étayer nos propositions à un troisième et dernier niveau, celui de l'utilisation. Les expériences présentées dans ce chapitre cherchent à montrer que les outils décrits permettent de mettre en place facilement des scénarios complexes tout en augmentant la qualité des résultats produits et en diminuant l'effort manuel nécessaire à leur production. Dans ce chapitre, nous parlerons d'effort de correction des erreurs, mais nous aborderons aussi l'effort de conception et expliquerons pourquoi il n'en est pas alourdi.

Avant de présenter l'articulation de ce chapitre, il est important de noter que plusieurs défis ont dû être surmontés pour permettre la réalisation des expériences que nous présentons, et l'évaluation des scénarios qu'elles mettent à l'épreuve. La manière de procéder est pourtant simple au niveau théorique : il faut se donner un problème d'interprétation de documents réaliste et comparer la quantité de travail manuel nécessaire entre un scénario de référence (imposant des corrections en post-traitement) et un scénario interactif (permettant la réintégration d'informations grâce à notre mécanisme itératif) pour atteindre un objectif de qualité donné.

En pratique, un certain nombre de difficultés rendent cette tâche délicate :

- les scénarios d'interprétation réels sont complexes et donc délicats à présenter (ils font intervenir de nombreux modules et échangent des données de plusieurs dizaines de types différents) ;
- la réalisation d'un système pour chaque scénario à évaluer risque de produire des systèmes comportant des différences dans la façon d'interpréter les pages, rendant alors difficile l'évaluation du scénario défini au niveau du module de stratégie globale ;
- la définition d'un objectif de qualité, et la mesure de la distance à cet objectif est délicate dans le cas de résultats structurés (comme des entrées d'une base de données de ventes, par exemple) ;
- l'évaluation du coût d'interaction est délicat car :

    1. la seule métrique indiquant précisément le coût d'un travail manuel est le temps, mais cette information dépend fortement de l'ergonomie des interfaces homme–machine mises en œuvre, et nous souhaitons limiter l'influence de ces éléments

dans l'évaluation d'un scénario car cela ne relève pas de notre travail ;

2. si on cherche à comptabiliser des actions abstraites (validation d'un cluster, étiquetage d'un mot, positionnement d'un numéro de vente), alors ces dernières constituent des grandeurs différentes et ne peuvent être comparées sans d'extrêmes précautions.

N'ayant pas trouvé dans la littérature de procédé détaillant l'évaluation de systèmes d'interprétation contextuelle ou assistée de fonds documentaires, nous avons mis en place une démarche spécifique afin de valider l'intérêt de scénarios tirant profit des mécanismes d'interaction que nous avons proposés. Cette démarche est basée sur les choix suivants.

1. Les scénarios comparés sont très simples, afin de limiter les définitions de données nécessaires, et de n'autoriser qu'un nombre très restreint d'actions de la part de l'opérateur humain. Ceci, permet d'éviter les problèmes de comparaison de grandeurs différentes ;
2. Les scénarios comparés sont basés sur la même description de page, et le scénario non interactif servant de référence correspond la première itération du scénario interactif. Ceci évite les biais dans la description des pages, et simplifie le développement.

Ce chapitre est alors organisé selon le plan suivant.

– La première expérience, présentée dans la section 8.1, met l'accent sur l'exploitation du contexte documentaire et présente un cas d'utilisation de l'interaction dirigée.
– La seconde expérience, présentée dans la section 8.2, se focalise sur la réintégration d'informations externes au niveau de la page et présente un cas d'utilisation de l'interaction spontanée.
– Nous terminons ce chapitre en présentant, en section 8.3, les enseignements que nous avons tirés de l'utilisation, en conditions réelles, de systèmes basés sur un mécanisme itératif.

8.1 Exploitation d'un contexte inter-pages et interaction dirigée

Dans cette section, nous décrivons une expérience montrant qu'il est possible alléger la charge de travail manuel nécessaire pour la transcription de patronymes dans des registres de ventes, en utilisant un mécanisme d'interaction dirigée qui permet d'exploiter des redondances entre ces mots.

Cette expérience compare deux scénarios.

Un *scénario de référence* basé sur un mécanisme d'interprétation *linéaire*. Il ne permet pas l'intégration de connaissances externe à la page lors de son interprétation. Il fait intervenir un module de stratégie globale *non itératif* et un module d'interprétation de page *non interactif*.

Un *scénario interactif* basé sur un mécanisme d'interprétation *itératif*. Il permet de tirer profit de connaissances externes à la page lors de son interprétation. Il fait intervenir un module de stratégie globale *itératif*, un module d'interprétation de page *interactif*, ainsi que deux modules permettant : (i) de regrouper des mots similaires, extraits de pages voisines, en clusters ; (ii) d'associer une étiquette à ces clusters.

8.1.1 Description du problème

Nous commençons par présenter le cas concret à la base de cette expérience, avant de décrire les simplifications réalisées pour l'expérience.

8.1.1.1 Situation réelle

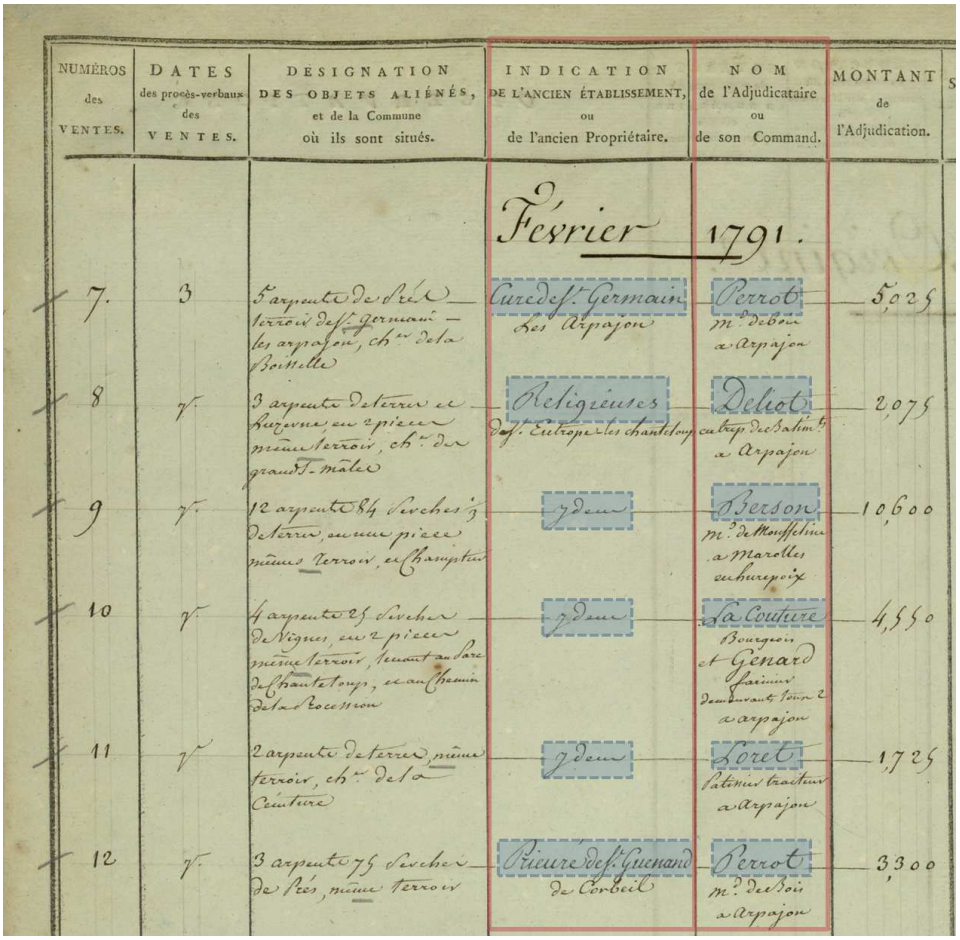


FIGURE 8.1 – Page de registre de ventes montrant les éléments à extraire. Les colonnes correspondant à l'ancien propriétaire et au(x) nouveau(x) propriétaire(s) sont encadrées en rouge, et les zones d'intérêt sont indiquées en bleu.

Le cas réel à la base de cette expérience est celui des registres de ventes auquel nous avons fait référence tout au long de ce manuscrit. La figure 8.1 montre un exemple de page où les colonnes contenant les éléments à localiser et reconnaître sont encadrées en rouge, et où les patronymes sont encadrés en bleu. Dans ces documents, les patronymes proviennent d'un lexique mal connu et ancien (pour ne pas dire « ouvert »), et la graphie des mots manuscrits les rend difficiles à lire (même pour un humain). Par ailleurs, le mot *« idem »* est utilisé pour éviter les recopies au sein d'une même page, ce qui impose de rechercher, pour la production des résultats finaux (ensembles d'informations classées par vente), le nom effectif de la personne désignée le mot *« idem »* .

Pour faciliter cette tâche, nous tirons profit de deux propriétés de ces documents historiques :

1. les ventes sont enregistrées par ordre chronologique pour une place d'enchères donnée (qui correspond à un ou plusieurs registre) ;
2. les ventes de biens d'anciens nantis (noblesse et clergé, principalement) sont effec-

tuées au sein d'une même période, et les acheteurs sont eux aussi actifs sur des périodes restreintes, si bien que les patronymes de chaque ancien propriétaire et de chaque nouvel acheteur sont mentionnés dans des ventes chronologiquement proches.

Il est alors possible de tirer profit de la redondance entre pages voisines de certains de ces mots pour les regrouper (avec un algorithme basé sur la ressemblance visuelle, différent de l'algorithme de classification) et augmenter la fiabilité de leur classification tout en permettant l'étiquetage d'un ensemble d'éléments avec une seule action de la part d'un opérateur humain.

8.1.1.2 Simplifications pour les besoins de l'expérience

Afin de pouvoir identifier précisément le gain apporté par l'exploitation du contexte documentaire formé par la redondance d'un patronyme dans un voisinage de pages, nous nous sommes concentrés sur l'évaluation de la classification des patronymes, en supposant qu'ils sont correctement localisés au préalable. Nous ne nous préoccupons pas non plus du problème de modification de l'hypothèse principale associée à un élément, qui peut nécessiter de modifier la structure résultat si cette hypothèse est *« idem »*, car le mécanisme d'interaction dirigée permet de gérer ce problème de façon transparente pour le concepteur du scénario.

Nous nous attachons dont à évaluer l'intérêt d'un scénario de regroupement de patronymes visuellement similaires (appelé *« scénario interactif »*), autorisant l'interaction avec un opérateur humain pendant la phase de traitement, par rapport à un scénario de référence où il faut corriger manuellement, en post-traitement, les étiquettes associées aux patronymes après leur reconnaissance isolée.

8.1.2 Scénario de référence (corrections en post-traitement)

Dans la mesure où on suppose que les zones correspondant à l'information de patronyme sont localisées au préalable, le travail du module d'interprétation de page se cantonne à invoquer un reconnaisseur adapté sur ces zones, et à renvoyer la liste des éléments reconnus. Aucune itération n'est possible dans ce cas, et l'interface homme–machine est sollicitée pour demander à un opérateur humain de valider ou corriger les mots pour lesquels l'indice de confiance n'est pas suffisant.

8.1.2.1 Conception des modules utilisés

Nous présentons ici la façon dont le module de stratégie globale et le module d'interprétation de page utilisés dans ce scénario sont conçus, et les données qu'ils échangent.

Module de stratégie globale La stratégie globale du système est la suivante : pour chaque page, le module de stratégie globale :

1. invoque le module d'interprétation de page et récupère la liste des éléments reconnus : pour chaque mot w_i, on dispose d'une liste de 10 hypothèses $h_{i,j} = (e_{i,j}, s_{i,j})$ (avec $j \in [1; 10]$) où $e_{i,j}$ est l'étiquette de l'hypothèse de reconnaissance j du mot w_i, et où s_{ij} est le score associé à cette hypothèse (la liste des hypothèses est triée par score décroissant) ;
2. pour chaque élément qui nécessite une correction manuelle, l'interface homme–machine est sollicitée pour demander à l'opérateur humain la valeur du mot.

La prise de décision entraînant la demande d'un étiquetage manuel dépend d'un seuil de rejet T_r. Ce seuil a été ajusté sur un ensemble de validation distinct des données expérimentales.

Module d'interprétation de page Au niveau du module d'interprétation de page, la description de la page contient, entre autres, la règle suivante pour une ligne de vente (exprimée en langage EPF simplifié où « + » indique un paramètre d'entrée, et « - » un paramètre de sortie) :

```
LigneVente(-LigneVente) ::=
   AT(zoneNumVente) &&
   detecterReconnaitreNombre(-NumVente) &&
   AT(zoneColProp) &&
   detecterReconnaitrePatronyme(-Prop) &&
   AT(zoneColAch) &&
   detecterReconnaitrePatronyme(-Ach) &&
   { contruireLigneVente +NumVente +Prop +Ach -LigneVente }
```

Cette description indique que pour construire une ligne de vente, il faut détecter et reconnaître son numéro, l'ancien propriétaire, et l'acheteur dans les colonnes appropriées.

Données échangées Dans ce scénario, les données échangées sont de la forme :

$$\begin{array}{c}(zone, \texttt{pat\_prop}, listeHypo)\\(zone, \texttt{pat\_ach}, listeHypo)\end{array} \tag{8.1}$$

Les deux types de données possibles concernent les patronymes de l'ancien propriétaire (`pat_prop`) et de l'acheteur (`pat_ach`), pour lesquels on dispose de la position et de la liste d'hypothèses renvoyée par le reconnaisseur.

8.1.2.2 Comportement du système pour le scénario de référence

Dans le cas de l'image de la figure 8.1, le module d'interprétation de page renverra le contenu indiqué dans l'encadré ci-après. Pour simplifier la lecture, on ne donne pas la valeur réelle des zones (coordonnées) : on note z_{Pi} la zone associée au patronyme de l'ancien propriétaire de la ligne i, et z_{Ai} la zone associée au patronyme de l'acheteur de la même ligne. Nous n'indiquons que le contenu de la mémoire correspondant aux trois premières lignes du tableau de la figure 8.1, en limitant l'écriture de la liste des hypothèses de chaque mot au meilleur élément.

Contenu de la mémoire visuelle *Après l'appel au module d'interprétation de page*

$$\{\ (z_{P1}, \texttt{pat\_prop}, [(\texttt{Cure}\ldots, 0,80),\ \ldots]),$$
$$(z_{A1}, \texttt{pat\_ach}, [(\texttt{Perrot}, \boxed{\mathbf{0,50}}),\ \ldots]),$$
$$(z_{P2}, \texttt{pat\_prop}, [(\texttt{Religieuses}, 0,90),\ \ldots]),$$
$$(z_{A2}, \texttt{pat\_ach}, [(\texttt{Deliot}, 0,60),\ \ldots]),$$
$$(z_{P3}, \texttt{pat\_prop}, [(\texttt{IDEM}, \boxed{\mathbf{0,40}}),\ \ldots]),$$
$$(z_{A3}, \texttt{pat\_ach}, [(\boxed{\texttt{Masson}}, \boxed{\mathbf{0,30}}),\ \ldots]),$$
$$\ldots\ \}$$

Observations : Le reconnaisseur appelé par le module d'interprétation de page a fait une erreur dans le dernier élément indiqué (`Masson` à la place de `Berson`, voir figure 8.1), et a une confiance inférieure au seuil T_r (indiquée en **gras** et encadrée) pour trois éléments.

Le scénario de référence sollicitera un opérateur humain pour les trois éléments incertains précédents.

8.1.3 Scénario interactif (regroupement et validation)

Dans ce scénario, nous souhaitons évaluer l'intérêt de regrouper les mots visuellement similaires entres pages voisines, à la fois pour fiabiliser les hypothèses de reconnaissance (en les fusionnant) et pour accélérer le travail d'étiquetage d'un opérateur humain. Le module d'interprétation de page est alors chargé de renvoyer l'ensemble des éléments reconnus dans la mémoire visuelle associée à l'image, et ensuite, au niveau global, la méthode de regroupement est appliquée avant de solliciter l'interface homme–machine pour les clusters de mots peu fiables.

On suppose toujours que les zones d'intérêt sont déjà localisées dans l'image, et même si nous conservons la possibilité de réintégrer l'information relative à la détection de mots *« idem »* , nous ne nous intéresserons pas à la valeur finale des patronymes dans les entrées relatives aux ventes, pour nous concentrer sur l'évaluation du scénario de regroupement.

Il est toutefois évident que cette mise en œuvre simplifie largement la modification de la structure résultat car la connaissance relative à l'organisation de la page reste au niveau du module d'interprétation de page. Il suffit de travailler sur les zones extraites au niveau global.

Ce scénario permet donc d'ajouter au scénario de référence présenté précédemment un comportement itératif permettant une amélioration progressive des résultats.

8.1.3.1 Conception des modules utilisés

Nous présentons ici la façon dont le module de stratégie globale, le module d'interprétation de page, le module de clustering (regroupement) et le module de reconnaissance des clusters (fusion d'hypothèses) utilisés dans ce scénario sont conçus, et les données qu'ils échangent.

Module de stratégie globale La stratégie globale de ce scénario étend celle du scénario de référence. Le module de stratégie globale est chargé de :

1. invoquer le module d'interprétation de page pour chaque page du lot, et de collecter les zones de patronymes extraites ;

2. invoquer un module de regroupement (clustering) basé sur une comparaison deux à deux des vignettes extraites qui produit des clusters de mots visuellement similaires ;
3. invoquer un module de fusion des hypothèses de reconnaissance pour les mots d'un même cluster ;
4. solliciter l'interface homme–machine pour les clusters peu fiables (il est seulement possible de donner une étiquette, pas de changer le cluster dans ce scénario) ;
5. invoquer à nouveau le module d'interprétation de page pour produire la structure résultat finale (structure tabulaire primitive).

Dans cette stratégie simplifiée pour les besoins de l'expérience, chaque page est traitée exactement deux fois par le module d'interprétation de page : une fois pour extraire les hypothèses de reconnaissance, et une fois pour réintégrer la valeur finale fournie par l'environnement. On pourrait parfaitement imaginer des méthodes avec plusieurs passes permettant une augmentation progressive de la qualité jusqu'à stabilisation.

Module d'interprétation de page Au niveau du module d'interprétation de page, la règle de description d'une ligne de vente est modifiée de la façon suivante (les ajouts sont indiqués **en gras**) :

```
LigneVente(-LigneVente) ::=
   AT(zoneNumVente) &&
   detecterReconnaitreNombre(-NumVente) &&
   AT(zoneColProp) &&
   EXTERNAL_CTRL(+pat_prop,
            detecterReconnaitrePatronyme, -Prop) &&
   AT(zoneColAch) &&
   EXTERNAL_CTRL(+pat_ach,
            detecterReconnaitrePatronyme, -Ach) &&
   { contruireLigneVente +NumVente +Prop +Ach -LigneVente }
```

Dans cette version, un opérateur `EXTERNAL_CTRL` spécial modifie le comportement du programme lors de la détection et la reconnaissance de chacun des patronymes (ancien propriétaire, et acheteur). Cet opérateur sert à poser systématiquement une question pour contrôler avec l'environnement la valeur d'un élément. Si une valeur a été fournie par l'environnement, alors elle est utilisée, sinon la meilleur hypothèse utilisée par le reconnaisseur est utilisée par défaut. Cet opérateur peut être mis en place de la façon suivante, à l'aide des opérateurs présentés en section 6.3 :

```
EXTERNAL_CTRL(+TypeDonnee, +Regle, -Resultat) ::=
   GET_ANSWER_OR_TRY(+TypeDonnee,
            ( Regle -Resultat &&
              CATCH_QUESTION(+TypeDonnee,
                        RAISE_QUESTION(zone_recherche, +TypeDonnee,
                                  { type_attendu: TypeDonnee,
                                    hypotheses: Resultat,
                                    ... }))))
```

La question est immédiatement interceptée pour permettre la poursuite de l'analyse, et est posée après avoir effectué la détection automatique. Si une information existe en mémoire visuelle, l'opérateur `GET_ANSWER_OR_TRY` englobant évite de refaire les calculs et de reposer une question. La question contient la liste des hypothèses générées par le reconnaisseur.

Module de clustering Le regroupement de mots visuellement similaires est réalisé en deux temps.

1. Pour chaque couple de vignettes de mots (w_i, w_j), on calcule une distance. On produit alors une matrice M où $M(i, j)$ représente la distance entre les mots w_i et w_j.
   Deux méthodes ont été expérimentées pour établir cette distance :
   – une méthode de programmation dynamique basée sur la technique DTW *(Dynamic Time Warping)* utilisant les caractéristiques au niveau signal de [101] ;
   – et une méthode basée sur l'appariement de points d'intérêt [15].
   Nous comparons ici les résultats obtenus avec la première méthode (publiés dans [43]) et ceux obtenus avec la seconde méthode.
2. Ensuite, un *clustering agglomératif* regroupe les mots jusqu'à une certaine limite. Il procède de la façon suivante :
   a) Initialiser l'opération en créant un cluster pour chaque mot.
   b) Agglomérer les clusters C_i et C_j pour chaque $i \neq j$ tant que $d_{i,j} \leq T_c$, avec

$$d_{i,j} = \min_{w_k \in C_i,\, w_l \in C_j} M(k, l) \tag{8.2}$$

Le seuil T_c a été ajusté sur un ensemble de validation distinct des données expérimentales.

Module de reconnaissance des clusters Pour chaque cluster C, les hypothèses $h_{i,j}$ de chaque mot $w_i \in C$ sont fusionnées de la façon suivante pour produire une nouvelle liste d'hypothèses $\hat{h}$ pour le cluster.
Soit

$$h' = \bigcup_{w_i \in C,\, j \in [1;10]} h_{i,j} \tag{8.3}$$

l'union des hypothèses pour tous les mots du cluster.
Pour chaque étiquette $\hat{e}$ possible dans cette ensemble d'hypothèses regroupées, on génère l'ensemble bien ordonné suivant :

$$\hat{h} = \left\{ (\hat{e}, \hat{s}) : \frac{1}{|h'|} \sum_{(e,s) \in h',\, e = \hat{e}} s \right\} \tag{8.4}$$

qui est trié selon les valeurs décroissantes des scores $\hat{s}$.

Comme dans le scénario de référence, un seuil de rejet T_r permet de déterminer si un cluster doit être considéré comme fiable ou non. Ce seuil a été ajusté sur un ensemble de validation distinct des données expérimentales.

Données échangées Par rapport au scénario de référence, deux formes de questions s'ajoutent aux données échangées entre le module d'interprétation de page et le module de stratégie globale. (On ne décrit pas ici le format des données échangées au niveau global.)

$$\begin{gathered}
(\mathit{zone}, \texttt{pat_prop}, \mathit{listeHypo}) \\
(\mathit{zone}, \texttt{pat_ach}, \mathit{listeHypo}) \\
(\mathit{zone}, \texttt{question}, \{\, \texttt{type_attendu}\colon \texttt{pat_prop}, \texttt{hypotheses}\colon \mathit{listeHypo} \,\}) \\
(\mathit{zone}, \texttt{question}, \{\, \texttt{type_attendu}\colon \texttt{pat_ach}, \texttt{hypotheses}\colon \mathit{listeHypo} \,\})
\end{gathered} \tag{8.5}$$

8.1.3.2 Comportement du système pour le scénario interactif

Le scénario interactif impose deux invocations du module d'interprétation de page. Lors de la première, il réclame une validation externe des éléments reconnus, et il réintègre ces informations lors de la seconde passe. (On ne décrit ici que les données échangées entre le module d'interprétation de page et le module de stratégie globale.)

Contenu de la mémoire visuelle *Après le 1er appel au module d'interprétation de page*

{ (z_{P1}, question { type_attendu: pat_prop,
hypotheses: [(Cure..., 0,80), ...] }),
(z_{A1}, question { type_attendu: pat_ach,
hypotheses: [(Perrot, **0,50**), ...] }),
(z_{P2}, question { type_attendu: pat_prop,
hypotheses: [(Religieuses, 0,90), ...] }),
(z_{A2}, question { type_attendu: pat_ach,
hypotheses: [(Deliot, 0,60), ...] }),
(z_{P3}, question { type_attendu: pat_prop,
hypotheses: [(IDEM, **0,40**), ...] }),
(z_{A3}, question { type_attendu: pat_ach,
hypotheses: [(Masson, **0,30**), ...] }),
... }

Observations : Le reconnaisseur appelé par le module d'interprétation de page a fait une erreur dans le dernier élément indiqué (Masson à la place de Berson, voir figure 8.1), et a une confiance inférieure au seuil T_r (indiquée en **gras** et encadrée) pour trois éléments. *Il a généré une question pour chacun des six éléments.*

Contenu de la mémoire visuelle *Avant le 2nd appel au module d'interprétation de page*

{ (z_{P1}, pat_prop, [(Cure..., 0,80), ...]),
(z_{A1}, pat_ach, [(Perrot, 1), ...]),
(z_{P2}, pat_prop, [(Religieuses, 0,90), ...]),
(z_{A2}, pat_ach, [(Deliot, 0,60), ...]),
(z_{P3}, pat_prop, [(IDEM, **0,80**), ...]),
(z_{A3}, pat_ach, [(Berson, 1), ...]),
... }

Observations : Après regroupement des éléments similaires, il a été possible d'augmenter automatiquement la confiance (en **gras**) du mot IDEM, et l'opérateur humain a été sollicité pour indiquer la valeur (encadrée) d'un cluster contenant le mot Perrot, et d'un autre contenant le mot (à présent corrigé) Berson.

8.1.4 Protocole expérimental

Après cette présentation des scénarios à comparer, nous présentons la mise en œuvre et les résultats du protocole expérimental permettant cette comparaison.

8.1.4.1 Jeu de données

Pour mener cette expérience, nous avons utilisé une base de plus de 11 000 vignettes de patronymes extraite des registres de ventes qui servent de cadre applicatif à notre travail. Ces vignettes sont extraites de plusieurs registres, et aucun filtrage n'a été réalisé. La redondance de chaque patronyme est variable, et le principal élément redondant correspond à *« idem »* avec plus de 1 500 régions concernées. Quelques exemples de vignettes de patronymes sont visibles en figure 8.2.

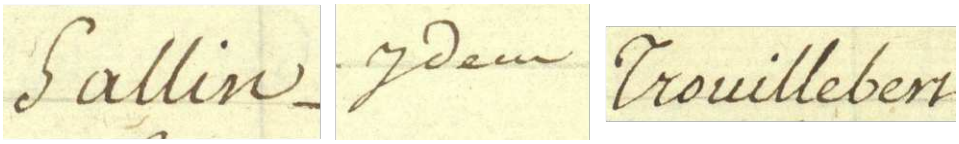


FIGURE 8.2 – Exemple de vignettes de patronymes contenues dans la base d'évaluation. De gauche à droite, on peut lire « Sallin », « idem » et « Trouillebert ».

8.1.4.2 Production des résultats pour les deux scénarios

Pour réaliser cette expérience, nous sommes partis du principe que le scénario de référence (sans clustering) est équivalent à la première étape du scénario interactif, c'est à dire à la première étape de reconnaissance des patronymes au sein du module d'interprétation de page. Dans la mesure où il est possible de récupérer l'information renvoyée par les reconnaisseurs dans les questions présentes dans la mémoire visuelle, nous avons lancé le scénario interactif et avons considéré les états suivants.

État initial Noté E_I, cet état correspond au moment où le scénario est démarré : aucun patronyme n'est reconnu, mais on suppose que la position de chacun de ces éléments est connue.

État après traitement automatique de référence Noté E_A, cet état correspond à l'instant du scénario où le module d'interprétation de page a été invoqué pour la première fois, et a généré des listes d'hypothèses pour chaque patronyme. Il correspond au résultat du scénario de référence pour lequel un fonctionnement itératif n'est pas possible.

État après traitement interactif expérimental Noté E_X, cet état correspond à l'instant du scénario où l'on s'apprête à réinjecter dans le module d'interprétation de page les nouvelles hypothèses correspondant à chacun des patronymes, après clustering, fusion des hypothèses, et sollicitation éventuelle d'un opérateur humain.

État après production de la vérité terrain Noté E_V, cet état correspond au résultat idéal et parfait d'un reconnaisseur pour chacun des patronymes. En pratique, il est généré grâce à une correction manuelle des données de l'état E_X.

La figure 8.3 illustre la production des différents états. Cette technique a permis : (i) de nous assurer que la description de page ne favorisait aucun scénario, puisqu'elle est ici unique ; (ii) de ne réaliser qu'un seul système pour les deux scénarios.

8.1.4.3 Méthode d'évaluation

Évaluation de la qualité de reconnaissance Pour évaluer la qualité de reconnaissance, nous avons regardé pour chaque vignette de patronyme des état E_A et E_X l'étiquette de

$$E_I \xrightarrow{\text{Scénario de référence}} E_A \xrightarrow{\text{Scénario interactif}} E_X \xrightarrow{\text{Corrections résiduelles}} E_V$$

FIGURE 8.3 – Génération des différents états, correspondant aux résultats des deux scénarios et à la vérité terrain, pour l'expérience de transcription de patronymes.

la meilleure hypothèse. Si cette dernière correspondait à la valeur attendue dans la vérité terrain (état E_V), alors nous comptions un résultat correct, sinon nous comptions une erreur.

Évaluation de la quantité de travail manuel La valeur des seuils T_r (rejet d'un hypothèse de reconnaissance jugée peu fiable, utile pour les deux scénarios) et T_c (limite d'agrégation de mots dans des clusters, utile dans le second scénario) permettent de régler la quantité de travail manuel qui sera nécessaire, en même temps que la qualité visée, car elles influent directement sur le taux d'erreur du système.

En réglant ces seuils, il est alors possible de fixer (empiriquement) un taux d'erreur, c'est à dire un objectif de qualité. Un objectif de 1% d'erreur pourra correspondre à un objectif de constitution d'une vérité terrain, tandis qu'un objectif aux alentours de 20% sera acceptable pour une tâche d'indexation où il suffit d'avoir la bonne hypothèse en début de liste des hypothèses, pour chaque élément.

Une fois que ce taux d'erreur est fixé, la quantité de travail manuel dépend directement de la fiabilité du processus de reconnaissance. On définit les compteurs suivants :

N_A Le **n**ombre de vignettes correctement annotées **a**utomatiquement.

N_M Le **n**ombre d'actions d'étiquetage **m**anuel. Si l'opérateur humain étiquette un cluster, cela compte pour une seule action.

N_E Le **n**ombre de vignettes pour lesquelles la meilleure hypothèse est **e**rronée.

En fonction de la fiabilité de chaque objet à reconnaître (vignettes isolées dans le scénario de référence, cluster dans le scénario interactif), ces derniers sont soit reconnus automatiquement, soit transmis à un opérateur humain pour étiquetage manuel.

Sont donc comptabilisées dans N_A :
- les vignettes correctement reconnues automatiquement ;
- et celles *correctement* étiquetées indirectement lorsque l'opérateur humain donne la valeur de l'étiquette d'un cluster.

On comptabilise dans N_M celles étiquetées *directement* par l'opérateur humain.

Concernant N_E, il comptabilise :
- les vignettes pour lesquelles la valeur attribuée automatiquement est erronée ;
- et les intrus dans cluster étiqueté par un opérateur humain.

Si E est l'ensemble des vignettes à étiqueter, alors on doit vérifier $|E| = N_A + N_M + N_E$.

On peut alors définir les taux suivants, qui serviront à donner les résultats de l'expérience :

T_A Le **t**aux d'étiquetage **a**utomatique indique la proportions d'éléments de l'ensemble E étiquetés automatiquement. Il est défini par :

$$T_A = 1 - \frac{N_M + N_E}{|E|} \tag{8.6}$$

T_M Le **t**aux d'étiquetage **m**anuel indique la proportions d'éléments de l'ensemble E étiquetés manuellement. Il est défini par :

$$T_M = \frac{N_M}{|E|} \tag{8.7}$$

T_E Le **t**aux d'**e**rreur indique la proportions d'éléments de l'ensemble E mal étiquetés. Il est défini par :

$$T_E = \frac{N_E}{|E|} \tag{8.8}$$

8.1.4.4 Résultats

La table 8.1 synthétise les résultats obtenus pour les deux scénarios, en prenant en compte les deux méthodes de calcul de distance (DTW et POI) dans le cas du second scénario. On n'indique que les taux d'erreur et d'étiquetage manuel, car le taux d'étiquetage automatique est liée aux deux autres.

TABLE 8.1 – Résultats comparés des scénarios pour l'expérience de transcription de plus de 11 000 zones de patronymes.

Taux d'erreur (T_E)	**Taux d'étiquetage manuel** (T_M)		
	Scénario de référence	Scénario interactif	
		Clustering DTW	Clustering POI
1%	75,8%	58,0%	55,6%
5%	50,8%	39,4%	36,5%
20%	20,2%	12,1%	10,7%

Ces résultats montrent que le système proposé permet de choisir simultanément le niveau de qualité souhaité et la quantité de travail manuel nécessaire. Pour un objectif de qualité très élevé (1% d'erreur), le système de référence impose de saisir près des trois quarts des étiquettes (75,8%), alors que la méthode la plus récente du système interactif permet de ne saisir qu'un peu plus de la moitié de ces éléments (55,6%). Pour un objectif d'indexation, où un taux d'erreur de 20% est acceptable, le taux d'étiquetage manuel passe de 20,2% à 10,7%, ce qui représente une réduction de près de la moitié des éléments à saisir.

Pour terminer cette section, on peut noter qu'un des avantages majeurs de notre approche est bien illustrée par le scénario interactif : il suffit de deux lignes supplémentaires dans la description de la page pour permettre la réintégration automatique de résultats produits de façon complexe au niveau global. Ceci permet d'utiliser au niveau de la page, et sans perturber la description de celle-ci, un *classifieur contextuel global*.

Un autre avantage majeur de notre approche est également illustré ici. Il s'agit de la possibilité de régler, au niveau global (ici à l'aide des seuils T_c et T_r), la quantité de travail manuel requise, ou la qualité des résultats visée, selon le point de vue.

On peut également noter que l'intégration de modules réalisés par différentes personnes a été facile à mettre en œuvre, et que la circulation d'information n'a pas été perturbée lors du replacement du module chargé de la comparaison des vignettes de mots.

Par ailleurs, il faut garder à l'esprit que, dans le cas réel, il est possible de visualiser les clusters de mots et d'éliminer des intrus. Bien que le coût de ces actions soit difficile à estimer, il est légitime d'espérer, en pratique, une réduction plus importante de la quantité de travail manuel requise pour la tâche de transcription présentée.

8.2 Gestion de cas de sous-segmentation avec l'interaction spontanée

Dans cette section, nous décrivons une expérience menée sur une tâche de correction de problème de sous-segmentation dans des énumérations de nombres, comme illustré en figure 8.4. Afin de faire face aux difficultés de ce type de document, une interaction *spontanée* est nécessaire, afin de permettre la détection et la correction des erreurs du système automatique par un opérateur humain.

De la même façon que dans la section précédente, cette expérience compare deux scénarios.

Un *scénario de référence* basé sur un mécanisme d'interprétation *linéaire*. Il ne permet pas l'intégration de connaissances externes à la page lors de son interprétation. Il fait intervenir un module de stratégie globale *non itératif* et un module d'interprétation de page *non interactif*.

Un *scénario interactif* basé sur un mécanisme d'interprétation *itératif*. Il permet de tirer profit de connaissances externes à la page lors de son interprétation. Il fait intervenir un module de stratégie globale *itératif*, un module d'interprétation de page *interactif*. Le mode d'interaction *spontanée* est utilisé, dans ce scénario, pour communiquer avec l'opérateur humain.

8.2.1 Description du problème

8.2.1.1 Situation réelle

Le document complet correspond à une « table » associée aux registres de ventes déjà présentés. Cette table sert à recenser, pour chaque commune et selon le type de vente, la liste des numéros des ventes concernées. Un exemple plus complet est visible en figure 8.5 à la page 160.

Une des principales difficultés auxquelles nous avons dû faire face dans l'interprétation de ces documents est l'effacement des séparateurs (virgules et *« à »* pour des valeurs contiguës) qui mène à plusieurs problèmes. Le plus courant est la fusion de deux nombres lors de la détection, ce qui pénalise tout ce qui suit dans le travail d'interprétation : reconnaissance, optimisation, etc. La figure 8.4 illustre ce problème.

La variabilité dans la taille des nombres et la présence de bruit rendent délicate la détection automatique de cas de sous-segmentation. Associés à la présence d'autres problèmes (sur-segmentation à cause du bruit en particulier), ces difficultés font du mode d'interaction *spontanée* une solution intéressante pour réduire le coût en correction, car il permet de corriger la cause des erreurs (l'absence de détection d'un séparateur par exemple) et de laisser le système reprendre automatiquement l'interprétation, plutôt que de corriger les conséquences (position et valeur des nombres).

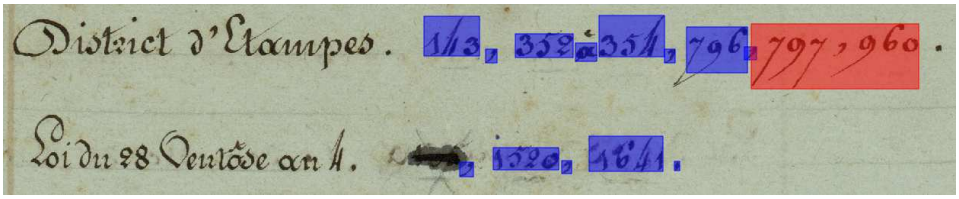


FIGURE 8.4 – Extrait de document montrant la détection automatique de séparateurs et de nombres. Un séparateur a été omis à la fin de la première ligne, causant la sous-segmentation des deux derniers nombres.

8.2.1.2 Simplifications pour les besoins de l'expérience

Afin de pouvoir identifier précisément le gain apporté par l'intégration d'une interaction spontanée dans le travail d'interprétation de ces énumérations de nombres, nous avons isolé un aspect précis du système réellement appliqué à ces documents. Nous avons extrait le mécanisme de correction de la sous-segmentation des nombres, c'est à dire ce qui permet, lorsqu'un opérateur humain détecte la fusion incorrecte de deux ou plusieurs numéros de vente, comme c'est le cas dans la figure 8.4, d'indiquer simplement la position du séparateur manquant. Ceci a pour effet de déclencher une nouvelle segmentation et évite à l'opérateur humain de devoir localiser précisément les nombres. On ne s'intéressera pas au résultats d'un reconnaisseur, pour centrer l'évaluation sur la qualité de localisation des éléments, et le gain dû à l'information apportée par le mode interactif.

Le scénario de référence de cette expérience est constitué d'une simple localisation, et impose la localisation précise des nombres mal segmentés lors d'une correction en post-traitement. Le scénario interactif, quant à lui, permet à l'opérateur humain de ne localiser que les séparateurs omis par le traitement automatique, dans la mesure où la segmentation peut être remise en cause.

8.2.2 Scénario de référence (corrections en post-traitement)

Dans ce scénario, on considère que pour chaque page, le module d'interprétation de page fournit la liste des positions des numéros de vente, et il ne peut être appelé à nouveau, car il ne dispose pas d'un mécanisme d'interprétation itératif : l'interface homme–machine est alors sollicitée et un opérateur humain doit :

1. localiser les problèmes de sous-segmentation ;
2. effacer les éléments erronés pour chaque cas ;
3. positionner précisément les zones correspondant aux numéros de vente.

Ce scénario est représentatif du travail à réaliser en post-traitement.

8.2.2.1 Conception des modules utilisés

Nous présentons ici comment le module de stratégie globale et le module d'interprétation de page utilisés dans ce scénario sont conçus, et les données qu'ils échangent.

Stratégie globale La stratégie globale est simpliste : pour chaque page, le module de stratégie globale :

1. appelle le module d'interprétation de page, qui renvoie les numéros localisés dans la mémoire visuelle [1] ;
2. sollicite l'interface homme–machine ;
3. conserve le résultat.

Module d'interprétation de page Au niveau du module d'interprétation de page, la description de la page est de la forme suivante (en langage EPF simplifié où « + » indique un paramètre d'entrée, et « − » un paramètre de sortie) :

```
ligneNumerosVentes() ::=
   detecterSeparateurs(-SepLst) &&
   extraireNombresEntreSeparateurs(+SepLst)

detecterSeparateurs(-[Sep|AutresSep]) ::=
   separateur(-Sep) &&
   detecterSeparateurs(-AutresSep)
detecterSeparateurs(-[])
```

Cette description indique que pour détecter une ligne de numéros de vente, il faut détecter les séparateurs, puis extraire les nombres présents entre ces séparateurs. La détection des séparateurs est une règle récursive qui détecte tous les séparateurs possibles (en consommant les primitives visuelles concernées) dans la zone correspondant à la ligne.

Données échangées Dans ce scénario, les données échangées sont de la forme :

$$(zone, \texttt{numero\_vente}, \_) \tag{8.9}$$

Le seul type de donnée possible concerne les numéros de ventes.

8.2.2.2 Comportement du système pour le scénario de référence

Dans le cas de l'image de la figure 8.4, le module d'interprétation de page renverra le contenu suivant. Pour simplifier la lecture, on ne donne pas la valeur réelle des zones (coordonnées) : on note alors z_N la zone associée à un nombre où N est la valeur du nombre (c'est arbitraire, car aucun mécanisme de reconnaissance n'est intégré dans ce scénario), ou « **797-960** » s'il s'agit du cas de sous-segmentation.

Contenu de la mémoire visuelle *Après l'appel au module d'interprétation de page*

{ $(z_{143}, \texttt{numero\_vente}, \_)$, $(z_{352}, \texttt{numero\_vente}, \_)$, $(z_{354}, \texttt{numero\_vente}, \_)$,
$(z_{796}, \texttt{numero\_vente}, \_)$, $\boxed{(z_{\textbf{797-960}}, \texttt{numero\_vente}, \_)}$, $(z_{1520}, \texttt{numero\_vente}, \_)$,
$(z_{1641}, \texttt{numero\_vente}, \_)$ }

Observations : Le module d'interprétation de page n'a pas pu détecter automatiquement le séparateur entre les deux derniers nombres de la première ligne. Il a fusionné les deux nombres et a produit un résultat incorrect (encadré) à cet endroit.

Il sera ensuite nécessaire de détecter et de corriger manuellement l'erreur de sous-segmentation.

1. Même si on n'utilise pas un fonctionnement itératif dans ce scénario, la mémoire visuelle facilite le transfert des données produites.

8.2.3 Scénario interactif (correction spontanée)

Dans ce scénario, on considère que pour chaque page, le module d'interprétation de page fournit la liste des positions des numéros de ventes (exactement comme dans le premier scénario) *et* la liste des séparateurs détectés. Ensuite, il est possible pour chaque page de venir corriger la détection de séparateur en ajoutant un élément manquant. Pour simplifier l'évaluation, on ne considère *que* les cas de sous-segmentation. L'interface homme–machine qui est alors sollicitée doit permettre à un opérateur humain de :

1. localiser les problèmes de sous-segmentation ;
2. effacer les éléments erronés pour chaque cas ;
3. positionner le (ou les) séparateur(s) manqué(s).

Ce scénario permet d'étendre le scénario de référence, en autorisant la réintégration d'informations externes après la première génération de résultats.

8.2.3.1 Conception des modules utilisés

Nous présentons ici comment le module de stratégie globale et le module d'interprétation de page utilisés dans ce scénario sont conçus, et les données qu'ils échangent.

Module de stratégie globale La stratégie globale correspond à la stratégie minimale nécessaire pour un fonctionnement compatible avec l'interaction spontanée : pour chaque page à traiter, le module de stratégie globale :

1. invoque le module d'interprétation de page sur cette page, puis récupère dans la mémoire visuelle la position des nombres localisés ;
2. sollicite l'interface homme–machine qui permet à un opérateur humain de détecter les erreurs, et proposer de nouveaux séparateurs ;
3. invoque à nouveau le module d'interprétation de page avec les informations fournies par l'opérateur humain, tant que des modifications ont été proposées par ce dernier.

Cette procédure est répétée tant que l'opérateur n'a pas validé la page.

Module d'interprétation de page Au niveau du module d'interprétation de page, la description de la page est de la forme suivante (les modifications par rapport à la version de référence sont indiquées **en gras**) :

```
LigneNumerosVente() ::=
   SPONTANEOUS(+sep_memoire, detecterSeparateurs(-SepLst)) &&
   extraireNombresEntreSeparateurs(+SepLst)

detecterSeparateurs(-[Sep|AutresSep]) ::=
   GET_ANSWER_OR_TRY(+sep_memoire, separateur(-Sep)) &&
   detecterSeparateurs(-AutresSep)

detecterSeparateurs(-[])
```

L'utilisation de l'opérateur `SPONTANEOUS` permet d'indiquer qu'il est possible d'ajouter ou modifier des données relatives aux séparateurs en mémoire pour chaque ligne. L'opérateur `GET_ANSWER_OR_TRY` indique, quant à lui, qu'il faut utiliser en priorité les séparateurs présents dans la mémoire visuelle à la position courante. Cette description permet d'utiliser les éléments éventuellement contenus dans la mémoire visuelle pour guider la détection des séparateurs et des numéros de vente.

Données échangées Dans ce scénario, les données échangées sont de la forme :

$$\begin{array}{c}(\textit{zoneNum}, \texttt{numero\_vente}, \_)\\ (\textit{zoneSep}, \texttt{sep\_memoire}, \_)\\ (\textit{zoneLigne}, \texttt{question\_optionnelle}, \{\, \texttt{type\_attendu: sep\_memoire} \,\})\end{array} \tag{8.10}$$

Deux nouveaux types de données peuvent être présents en mémoire : les séparateurs, et les questions implicites liées à la gestion de l'interaction spontanée.

8.2.3.2 Comportement du système pour le scénario interactif

Dans le cas de l'image de la figure 8.4, l'évolution du contenu de la mémoire visuelle est le suivant :

Contenu de la mémoire visuelle *1re itération, après l'appel au module d'interprétation de page*

$\{\, (\textit{zLigne}_1, \texttt{question\_optionnelle}, \{\, \texttt{type\_attendu: sep\_memoire} \,\}),$
$(\textit{zNum}_{143}, \texttt{numero\_vente}, \_),\ (\textit{zSep}_{11}, \texttt{sep\_memoire}, \_),$
$(\textit{zNum}_{352}, \texttt{numero\_vente}, \_),\ (\textit{zSep}_{12}, \texttt{sep\_memoire}, \_),$
$(\textit{zNum}_{354}, \texttt{numero\_vente}, \_),\ (\textit{zSep}_{13}, \texttt{sep\_memoire}, \_),$
$(\textit{zNum}_{796}, \texttt{numero\_vente}, \_),\ (\textit{zSep}_{14}, \texttt{sep\_memoire}, \_),$
$\boxed{(\mathbf{zNum_{797\text{-}960}}, \texttt{numero\_vente}, \_),}$
$(\textit{zLigne}_2, \texttt{question\_optionnelle}, \{\, \texttt{type\_attendu: sep\_memoire} \,\}),$
$(\textit{zSep}_{21}, \texttt{sep\_memoire}, \_),\ (\textit{zNum}_{1520}, \texttt{numero\_vente}, \_),$
$(\textit{zSep}_{22}, \texttt{sep\_memoire}, \_),\ (\textit{zNum}_{1641}, \texttt{numero\_vente}, \_),$
$(\textit{zSep}_{23}, \texttt{sep\_memoire}, \_)\ \}$

Observations : Le module d'interprétation de page n'a pas pu détecter automatiquement le séparateur entre les deux derniers nombres de la première ligne. Il a fusionné les deux nombres et a produit un résultat incorrect (encadré) à cet endroit. *Il a généré des questions optionnelles pour chaque ligne, autorisant des modifications externes spontanées.*

1re itération,

Contenu de la mémoire visuelle *après l'appel à l'interface homme–machine*

{ ($zLigne_1$, question_optionnelle, { type_attendu: sep_memoire }),
($zNum_{143}$, numero_vente, _), ($zSep_{11}$, sep_memoire, _),
($zNum_{352}$, numero_vente, _), ($zSep_{12}$, sep_memoire, _),
($zNum_{354}$, numero_vente, _), ($zSep_{13}$, sep_memoire, _),
($zNum_{796}$, numero_vente, _), ($zSep_{14}$, sep_memoire, _),
(zSep$_{15}$, sep_memoire, _),
($zLigne_2$, question_optionnelle, { type_attendu: sep_memoire }),
($zSep_{21}$, sep_memoire, _), ($zNum_{1520}$, numero_vente, _),
($zSep_{22}$, sep_memoire, _), ($zNum_{1641}$, numero_vente, _),
($zSep_{23}$, sep_memoire, _) }

Observations : L'opérateur humain a supprimé le numéro de vente erroné (en position $zNum_{797\text{-}960}$), et a indiqué la position du séparateur manqué (en position **zSep$_{15}$**).

2de itération,

Contenu de la mémoire visuelle *après l'appel au module d'interprétation de page*

{ ($zLigne_1$, question_optionnelle, { type_attendu: sep_memoire }),
($zNum_{143}$, numero_vente, _), ($zSep_{11}$, sep_memoire, _),
($zNum_{352}$, numero_vente, _), ($zSep_{12}$, sep_memoire, _),
($zNum_{354}$, numero_vente, _), ($zSep_{13}$, sep_memoire, _),
($zNum_{796}$, numero_vente, _), ($zSep_{14}$, sep_memoire, _),
(zNum$_{797}$, numero_vente, _), ($zSep_{15}$, sep_memoire, _),
(zNum$_{960}$, numero_vente, _),
($zLigne_2$, question_optionnelle, { type_attendu: sep_memoire }),
($zSep_{21}$, sep_memoire, _), ($zNum_{1520}$, numero_vente, _),
($zSep_{22}$, sep_memoire, _), ($zNum_{1641}$, numero_vente, _),
($zSep_{23}$, sep_memoire, _) }

Observations : Le module d'interprétation de page a automatiquement détecté la position précise des nombre sous-segmentés auparavant, aux positions **zNum$_{797}$** et **zNum$_{960}$**, grâce au séparateur indiqué par l'opérateur humain.

Ensuite, ce nouvel ensemble de résultats sera validé à l'aide de l'interface homme–machine par un opérateur humain, et le traitement de cette image sera terminé.

8.2.4 Protocole expérimental

Après cette présentation des scénarios à comparer, nous présentons la mise en œuvre et les résultats du protocole expérimental permettant cette comparaison.

8.2.4.1 Jeu de données

Pour mener cette expérience, nous nous sommes basés sur 50 images prises aléatoirement dans le fonds fourni par les archives départementales des Yvelines (voir sous-section 1.1.2.2). Ces images sont similaires à celle présentée en figure 8.5. Ces 50 images contiennent 1 637 numéros de vente à localiser.

8.2.4.2 Production des résultats pour les deux scénarios

Nous avons généré et enregistré l'état de la mémoire visuelle associée à chaque image du jeu de données à plusieurs instants au cours de l'expérience.

État initial Noté E_I, cet état correspond à un instant de l'interprétation du lot d'images où, pour chaque page, on connaît la position de chaque ligne et il ne reste plus qu'à détecter la position des nombres.

État après traitement automatique de référence Noté E_A, cet état correspond à l'instant de l'interprétation où, à partir de E_I, l'application du scénario de référence a produit un ensemble de résultats localisant les numéros de vente de façon complètement automatique, qu'il faut ensuite corriger.

État après traitement interactif expérimental Noté E_X, cet état correspond à l'instant de l'interprétation où l'application du scénario interactif a permis la production d'un ensemble de résultats en interaction avec un opérateur humain. Dans la mesure où l'état E_A correspond à la première moité de la première itération du scénario interactif (première génération de résultats sans assistance humaine), nous avons choisi de générer E_A en utilisant le scénario interactif, car cette opération n'a aucun coût humain, et cela permet par ailleurs d'utiliser exactement la même description au niveau de la page, ce qui est plus rigoureux (on ne fait varier que le scénario).

État après production de la vérité terrain Noté E_V, cet état correspond au résultat idéal attendu pour la détection des numéros de vente, et est généré grâce à une correction et complétion manuelle des résultats obtenus à l'état E_X. Ces corrections incluent le positionnement d'éléments impossibles à détecter avec la description de référence : cas de sur-segmentation, ajouts de numéros entre les lignes, gestion du bruit, etc.

La figure 8.6 illustre la production des différents états.

8.2.4.3 Méthode d'évaluation

L'enjeu de l'évaluation est de déterminer :

1. le gain en qualité obtenu grâce à l'application du scénario interactif, c'est à dire l'augmentation du nombre d'éléments bien détectés, et la diminution du nombre d'éléments mal détectés, entre l'état E_A et l'état E_X, relativement à l'état E_V (qui permet de vérifier que ce gain est significatif) ;
2. la coût en travail manuel (pour l'opérateur humain) nécessaire pour passer de l'état E_I à l'état E_X, ce qui correspond au coût pour passer de l'état E_A à l'état E_X ; le passage de l'état E_I à l'état E_A étant complètement automatique.

Pourquoi n'avons-nous pas comparé le coût d'un passage de l'état E_I à l'état E_V pour les deux scénarios ? La raison est simple : dans la mesure où, dans un cas réel, il est possible d'ajouter d'autres mécanismes pour atteindre une qualité plus élevée avec une approche assistée par l'opérateur humain, il nous est apparu plus représentatif de l'intérêt de nos

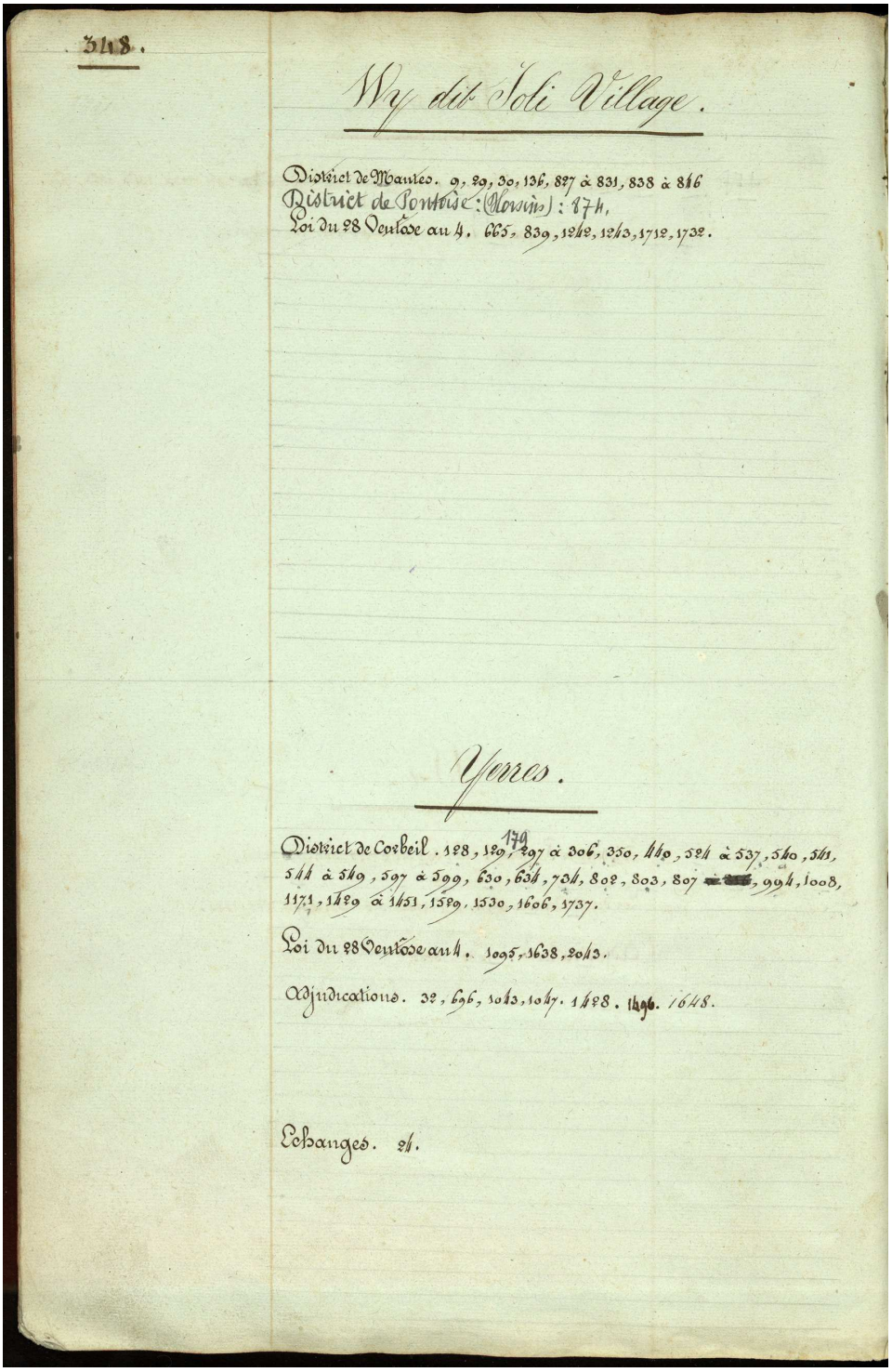


FIGURE 8.5 – Exemple de page de *table des ventes*, extraite du fonds documentaire du XVIII^{e} siècle que nous avons traité.

$$E_I \xrightarrow{\text{Scénario de référence}} E_A \xrightarrow{\text{Scénario interactif}} E_X \xrightarrow{\text{Corrections résiduelles}} E_V$$

FIGURE 8.6 – Génération des différents états pour l'expérience de correction de cas de sous-segmentation grâce à l'interaction spontanée.

travaux d'estimer la quantité de travail qu'il aurait fallu pour atteindre l'état E_X produit de façon assistée si on devait corriger l'état E_A avec une méthode classique en post-traitement.

Estimation de la qualité de la localisation Pour estimer la qualité relative à l'état idéal de la vérité terrain E_V, nous avons cherché à identifier dans les états E_A et E_X trois types de zones de numéros d'ordre.

Les zones correctement localisées Ces zones sont celles qui appartiennent à E_A (respectivement E_X) pour lesquelles nous considérons qu'il y a correspondance avec une zone de E_V (il y a unicité car la vérité terrain ne permet pas de recouvrements).

Les zones non détectées (ou « manquantes ») Ces zones sont celles qui sont attendues dans la vérité terrain E_V mais pour lesquelles aucun élément n'est présent dans E_A (respectivement E_X).

Les zones de bruit Ces zones sont celles qui sont présentes dans E_A (respectivement E_X), pour lesquelles aucun élément n'est présent dans la vérité terrain E_V.

Pour déterminer si une zone détectée (notée z_D) dans un premier état correspond à une zone attendue (notée z_A) dans un second état, nous comparons deux à deux les zones des deux états. Pour être identifiées comme *correspondantes*, deux zones z_D et z_A doivent vérifier :

$$\frac{surface(z_D \cap z_A)}{surface(z_A)} > T \quad \wedge \quad \frac{surface(z_D \cap z_A)}{surface(z_D)} > T \tag{8.11}$$

où $surface(Z)$ correspond au nombre de pixels dans la zone Z, et où $T \in [0;1]$ est un seuil qui indique le taux de recouvrement que doit avoir l'intersection des zones par rapport à chacune des zones.

Nous avons utilisé une valeur de 0.80 pour T afin de tolérer une certaine variation dans la forme des zones qui peuvent varier légèrement à cause de la taille des traits descendants de certains chiffres, comme les *« 7 »* ou les *« 9 »* , et pour lesquelles la partie basse ne porte que peu d'information.

L'estimation de la qualité d'un état est donc donnée par la quantité de zones correctes, manquantes et de bruit de cet état, par rapport à la vérité terrain.

Estimation de la quantité de travail manuel Afin de ne pas rendre l'évaluation des scénario dépendant de la qualité de l'interface homme–machine proposée, nous basons l'estimation du coût de correction manuel sur le nombre d'opérations à réaliser.

Pour le scénario automatique, le travail manuel en post-traitement nécessite pour chaque erreur :

1. de la repérer dans l'image ;
2. de supprimer les éléments erronés ;
3. *de localiser précisément deux numéros de ventes ou plus*.

Pour le scénario interactif, le travail manuel en cours de traitement nécessite pour chaque erreur :

1. de la repérer dans l'image ;
2. de supprimer les éléments erronés ;
3. *de localiser un ou plusieurs séparateurs manquants*.

Bien que le travail de correction soit en pratique plus rapide dans le cas du scénario interactif (il n'est pas gênant de supprimer des éléments voisins s'ils sont détectés automatiquement, et le séparateur peut être localisé avec un simple point), nous avons fait le choix de considérer que la localisation d'un élément avait le même coût dans les deux cas.

Les étapes préliminaires de repérage et de suppression des éléments erronés étant identiques dans les deux scénarios, nous avons compté le nombre d'éléments ajoutés (ou qu'il aurait fallu ajouter) dans chacun des cas pour déterminer le coût associé.

8.2.4.4 Résultats

L'évaluation des données expérimentales a produit les résultats suivants.

Amélioration de la qualité La variation de qualité (relative à la vérité terrain E_V) entre les états E_A (obtenu avec le scénario de référence) et E_X (obtenu avec le scénario interactif) sont synthétisés dans la table 8.2.

TABLE 8.2 – Résultats comparés des scénarios pour l'expérience de segmentation de nombres dans 50 images contiennent 1 637 numéros de vente. L'état E_A correspond aux résultats du scénario de référence, et l'état E_X à ceux du scénario interactif.

Zones	E_A	E_X	**Variation**	
Correctes	1339	1460	+121	(+7,5%)
Manquées	298	177	-121	(-40,6%)
Bruit	233	169	-64	(-27,5%)
Bruit/Correctes	17,4%	11,6%	-5,8	(-33,3%)

Ces chiffres montrent un gain notable en terme de zones correctement reconnues (+7,5%), ce qui correspond à plus de 40% des éléments manqués. De plus, la quantité de zones de bruit est considérablement diminuée (-27,5%) grâce à l'apport d'informations structurantes : une connaissance fiable de la position des séparateurs améliore largement la détection des numéros de ventes, et évite des fragmentations avec l'algorithme utilisé. Cette diminution des éléments perturbateurs correspond à un tiers du bruit relatif (rapport zones de bruits / zones correctes).

Réduction du coût Le scénario interactif qui a permis la génération de l'état E_X a nécessité la saisie de *85 nouveaux séparateurs*, ce qui a entraîné la détection de 121 numéros de vente supplémentaires. Il peut être noté que ce nombre de 85 n'est pas nécessairement minimal, car certaines actions d'édition peuvent ne pas avoir suffi pour permettre une détection correcte des numéros de vente avoisinants, à cause de dégradation de ces numéros, par

exemple. La table 8.3 compare pour les deux scénarios les actions requises, et leur nombre, pour atteindre le niveau de qualité de l'état E_X (en nombre de zones correctement détectées, en négligeant la suppression du bruit).

TABLE 8.3 – Estimation du coût en travail manuel de chacun des scénarios pour passer de l'état E_I à l'état E_X.

Scénario	**Type d'action**	**Nombre d'actions**
Automatique	Positionnement de numéros	121
Interactif	Positionnement de séparateurs	85

En nombre d'actions nécessaires, le scénario interactif a permis de *diminuer de 29,8%* le nombre d'actions nécessaires pour atteindre le niveau de qualité de l'état E_X.

Pour conclure cette section, on peut rappeler que la nécessité d'isoler un échange d'informations très précis dans un cas réel a été indispensable pour permettre une évaluation rigoureuse de la qualité des résultats produits et du coût associé ; en évitant de mélanger différentes actions du côté de l'opérateur humain, et en minimisant l'influence de l'ergonomie de l'interface homme–machine.

Bien que l'expérience décrite présente un cas très basique, elle montre déjà un gain au niveau de la qualité des résultats produits et une diminution de la quantité de travail manuel à réaliser. Ces avantages sont visibles, bien que le protocole de test et la méthode d'évaluation favorisent le scénario de référence à deux niveaux :

- ils minimisent le coût de localisation de numéros de vente ;
- ils ne considèrent pas la nécessité, en pratique, de produire un résultat structuré et donc d'être capable d'identifier le nom de commune et le type de vente auxquels un numéro de vente doit être associé.

De plus, le cas présenté a été choisi pour la simplicité des scénarios et des descriptions de page nécessaires, ce qui impose une relation de dépendance relativement directe entre les résultats remis en cause (la localisation des numéros de vente) et les données fournies par l'environnement (la position des séparateurs manquants). Il est parfaitement possible d'envisager des situations où l'environnement remet en cause une information (comme la position des bords d'un tableau) qui aura des conséquences beaucoup plus lourdes sur le reste de l'interprétation, et produira un gain beaucoup plus élevé.

8.3 Conception et exploitation du système en conditions réelles

Cette section a pour objectif de donner au lecteur la possibilité d'apprécier les conséquences et l'intérêt de l'utilisation réelle d'un système fonctionnant selon un mécanisme itératif.

Nous partons d'un exemple de cas difficile pour illustrer l'intérêt de cette approche au niveau de la conception de la description d'une page, et mettre en évident la façon dont il est possible de faire face à des situations délicates. Nous évoquons finalement la particularité du fonctionnement, par paliers, que nous avons pu constater dans un un tel système.

8.3.1 Exemple d'image complexe

Malgré les efforts qu'il peut consacrer à la description précise des configurations possibles des contenus des différentes images d'un fonds documentaire, le concepteur d'un système de traitement de documents en grand volume ne peut anticiper toutes les situations auxquelles le système devra faire face. Nous raisonnons fréquemment sur des images telles que celle de la figure 8.7, et nous retrouvons généralement face à des images telles que celle de la figure 8.8.

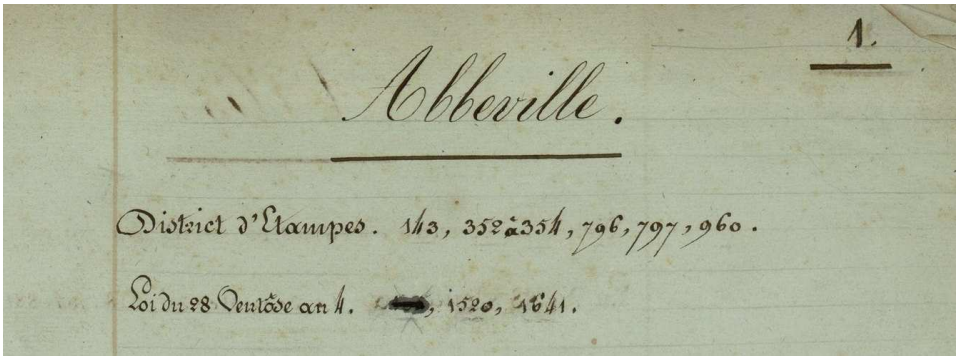


FIGURE 8.7 – Exemple d'image « facile ».

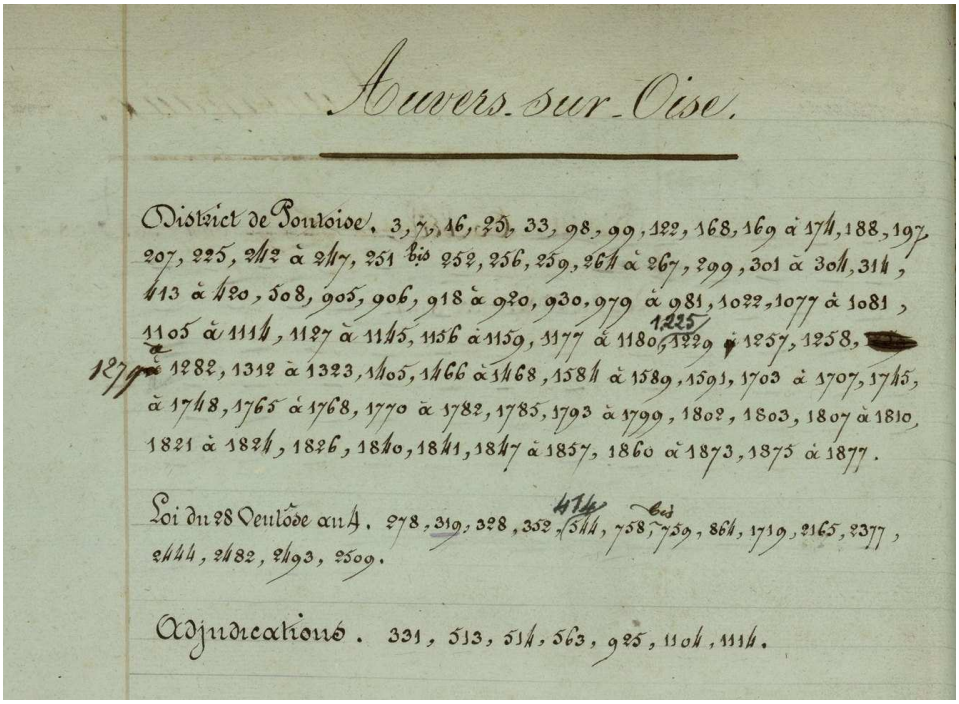


FIGURE 8.8 – Exemple d'image « difficile ».

L'image de la figure 8.8 montre certaines difficultés typiques qu'on peut facilement oublier, ou ne pas avoir anticipé :

- la structure du contenu s'étend sur plusieurs lignes ;
- des ratures, des effacements et des recouvrements entre traits de différentes lignes perturbent la détection du tracé ;
- des rajouts de numéros entre les lignes, ou hors des lignes, rendent la structure complexe ;
- de nouveaux éléments comme un *« bis »* nécessitent de considérer des éléments qui ne sont plus des nombres…

Face à ces problèmes, il est délicat de savoir quel effort de conception est réellement nécessaire, car la gestion de cas particulier peut rapidement devenir extrêmement coûteuse, pour un gain en qualité dérisoire *in fine*.

8.3.2 Simplification de la conception de la description de page

Un avantage pratique immédiat de l'utilisation d'un système permettant une interprétation itérative est de limiter, au moins dans un premier temps, la difficulté de l'étape de conception. Grâce à la possibilité de tolérer l'imprévu ou l'incertain, il n'est plus nécessaire de faire la chasse aux cas particuliers pour garantir des résultats corrects, à condition de laisser certains degrés de liberté à la description de page.

En pratique, la capacité à conserver les résultats intermédiaires ou provisoires facilite largement la mise au point. Il est également moins fréquent, lors du développement, de se surprendre à consacrer une journée de développement à régler des cas qui n'apparaissent que très ponctuellement dans le fonds considéré.

Bien qu'il soit très difficile d'estimer l'effort de conception, il nous a semblé que cette approche itérative diminuait la crainte des cas mal gérés, et incitait à se focaliser d'abord sur les cas généraux qu'il est alors plus facile d'identifier en lançant le système sur des petits volumes. La gestion des cas problématiques pourra aussi être effectuée (dans certains cas) après une phase de traitements en grand volume, car le système aura conservé les résultats corrects et facilitera le traitement des pages problématiques séparément.

Finalement, la gestion de documents en petits volumes peut également être un domaine d'application des systèmes itératifs, car ils permettent de n'implémenter que le minimum de modules automatiques pour réaliser à la main les traitements restants, tout en conservant les informations fournies et en autorisant des modifications dans les modules utilisés.

8.3.3 Utilisation d'un mode d'interaction hybride

Les degrés de liberté qu'il faut réussir à intégrer dans la description d'une page dépendent d'une part de l'anticipation du concepteur, et de sa capacité à mettre en place un mécanisme de détection d'erreur. Si la détection automatique d'erreur n'est pas possible, le mécanisme d'interaction spontanée permet de disposer d'une solution de repli, mais il est coûteux de laisser un opérateur humain passer en revue les pages à la recherche d'erreurs.

En pratique, cette détection n'est parfois possible que partiellement : il est possible de s'apercevoir que les numéros de ventes qu'on essaie de lire dans une colonne ont tous une fiabilité très faible, et on peut alors suspecter que la colonne détectée n'est pas la bonne. La question devient alors imprécise : « La colonne de numéros d'ordre est-elle bien à cette position ? Sinon où est-elle ? » De plus, il est fréquent qu'un opérateur humain repère des erreurs non détectées automatiquement en corrigeant une erreur détectée automatiquement.

Qu'en est-il des éléments oubliés ? Si dans une séquence d'éléments un numéro de ventes est manqué, comment l'ajouter ? Dans des descriptions réelles, le mode d'interaction

utilisé est une hybridation des modes dirigé et spontané. Il peut être possible de répondre plusieurs fois à une question, de modifier la position de certains éléments sans question préalable, de préciser la zone de mots détectés pour faciliter leur reconnaissance, etc.

Par ailleurs, beaucoup de données de types variés (position et valeur de numéros d'ordre, position de colonnes, etc.) circulent entre les modules du système à tout moment.

Nous avons commencé à définir certains degrés de liberté dans l'information qu'il est possible de fournir au système, comme avec la zone d'acceptation des réponses, ou avec des contraintes sur le domaine de valeur de la réponse à une question. Les problèmes de multiplicité des réponses, et la composition des modes d'interaction sont encore l'objet de travaux.

8.3.4 Comportement par paliers d'un système itératif en fonctionnement

Lors du fonctionnement d'un système itératif, certains problèmes ne pourront pas être gérés, malgré les degrés de liberté qu'il est possible de laisser dans la description à l'aide de l'extension du langage que nous avons proposée.

Il est évident que des cas qui ne sont pas prévus par la description d'origine, et pour lesquels les degrés de liberté laissés dans cette description ne seront pas suffisants pour les tolérer.

Pour une tâche donnée (par exemple la localisation de numéros de ventes de l'expérience de la section 8.2), on peut typiquement distinguer quatre paliers de qualité des résultats dans l'interprétation d'un fonds documentaire.

1. Le premier palier est atteint lorsque l'interprétation automatique a proposé ses premiers résultats, normalement après le premier appel au module d'interprétation de page lors de la première itération globale. Ce palier est théoriquement la limite des méthodes automatiques (considérant la page séparée de son environnement) pour le problème concerné.
   Nous proposons d'appeler ce palier la **limite automatique**.
2. Le deuxième palier est atteint lorsque tous les problèmes détectés automatiquement ont trouvé des réponses. Ce palier est la limite de qualité des résultats des approches basées uniquement sur une interaction dirigée.
   Nous proposons d'appeler ce palier la **limite de détection**.
3. Le troisième palier est atteint lorsqu'il n'est plus possible de fournir spontanément de nouvelles informations au système qui soient prises en compte par le module d'interprétation de page, car sa description ne permet plus de représenter ces éléments. C'est par exemple le cas des nombres écrits entre les lignes dans la figure 8.8.
   Nous proposons d'appeler ce palier la **limite de la description**.
4. Le quatrième palier correspond à la **vérité terrain**. Il n'est possible de l'atteindre qu'en désactivant le mécanisme itératif pour les pages concernées, et de basculer dans une édition manuelle. La structuration des résultats peut rendre cette édition complexe, mais normalement les interfaces graphiques nécessaires à l'interaction ont requis la mise en place de fonctionnalités proches des actions d'éditions, tant dans la représentation des données et leur manipulation que dans l'interprétation de celles-ci.

Généralement, il est possible de répondre aux questions et de fournir spontanément d'autres informations pour accélérer l'accès au troisième palier depuis le premier, et éviter une progression en deux parties plus lente.

Dans le cas des tables de ventes, illustrées partiellement en figure 8.7 et figure 8.8, la description réellement utilisée localisait les lignes de texte avant d'extraire les nombres. Elle posait des questions à propos du type de ligne (numéros seulement, ou texte puis numéros) avant d'en extraire le contenu. Elle tolérait l'ajout et la suppression (dans une certaine mesure) de séparateurs et de nombres.

Nous avons chronométré approximativement le temps de travail d'un opérateur humain (l'auteur) nécessaire à la réalisation des tâches suivantes :

1. réponse aux questions relatives au type de ligne : c'est une phase d'interaction dirigée ;
2. correction des cas de sous-segmentation (ajout de séparateurs manquants) : c'est une phase d'interaction spontanée dont l'objectif et le périmètre sont clairement définis ;
3. correction de tous les autres problèmes de segmentation de nombres (éléments oubliés, bruit détecté à tort, etc.) : c'est une phase d'interaction spontanée aux objectifs plus flous ;
4. correction manuelle après désactivation du mécanisme itératif pour les derniers éléments non acceptés par la description.

Le graphique de la figure 8.9 donne une vision globale de l'évolution de la qualité par rapport au coût réel du travail manuel (dépendant de l'ergonomie de l'interface homme–machine). Il ne respecte pas l'échelle afin de faire apparaître les éléments intéressants. Il montre également le comportement possible d'un scénario réel, qui mélangerait interaction spontanée et interaction dirigée. Ce scénario permettrait d'atteindre plus rapidement une meilleure qualité, mais s'arrêterait avant d'atteindre la limite de la description, pour éviter une explosion du coût.

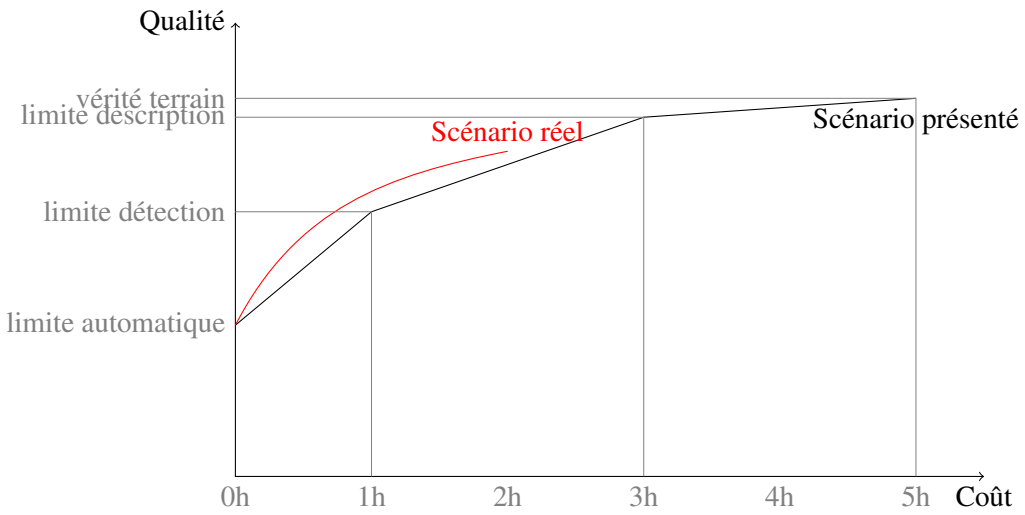


FIGURE 8.9 – Représentation approximative de l'évolution de la qualité des résultats par rapport au coût manuel, pour le scénario interactif présenté, et pour un scénario réel.

Concernant la désactivation du mécanisme itératif, elle est faite page par page et peut être :

– explicite, c'est à dire demandée par un opérateur humain ;

– implicite, c'est à dire provoquée par un opérateur humain (lorsqu'il n'effectue aucune modification sur une page et passe à la suivante, par exemple) ;
– automatique, c'est à dire provoquée par le système (avec une limite d'itérations, ou un critère de qualité, par exemple).

En conclusion, nous avons montré dans cette section les avantages pratiques des outils proposés, et leur comportement réel. Le potentiel de ces outils est important, mais il doit cependant être mieux appréhendé, appelant alors à de nouveaux travaux.

8.4 Conclusion

Ce chapitre a montré que la mise en place de scénarios complexes pouvait être facilement réalisée à l'aide des outils que nous proposons, et que les systèmes spécifiques ainsi construits présentaient des performances très intéressantes. La première expérience, portant sur plus de 11 000 zones de patronymes (voir sous-section 8.1.4.4, page 152), montre qu'il est possible, en ajoutant quelques lignes dans la description, de bénéficier, au niveau de la page, d'un *classifieur contextuel global* présentant des performances bien meilleures que celles d'un classifieur considérant les données séparément. Cette expérience a également montré qu'il est possible de maîtriser le compromis entre la qualité des résultats et le coût de production de ces derniers. La seconde expérience, portant sur 1 637 numéros de vente (voir sous-section 8.2.4.4, page 162), a confirmé la possibilité d'augmenter la qualité des résultats en limitant le nombre de zones de bruits détectées, tout en diminuant la quantité d'actions manuelles requises.

Ces nouveaux outils permettent alors une conceptions plus rapide d'un système. Ils limitent l'impact des cas particuliers car ils permettent de tirer profit des corrections effectuées par les opérateurs humains même si aucune erreur n'a pu être détectée. Il facilitent également l'intégration de différentes modules pour la mise en œuvre d'un scénario efficace.

Finalement, l'utilisation de ce type de système, basé sur un mécanisme d'interprétation itératif, dans des situations réelles, nous a permis de mieux comprendre la façon dont nous pouvons l'utiliser, ainsi que les particularités de son fonctionnement. Nous avons constaté, en pratique, que le mode d'interaction effectif était un compromis entre un mode complètement dirigé (où l'identification des problèmes est précise) et un mode complètement spontané (où on laisse l'opérateur humain complètement libre). Par ailleurs, nous avons remarqué que le comportement à l'exécution de ce type de système présente un phénomène de progression par paliers qui ouvre des pistes de recherche intéressantes.

Quatrième partie

Conclusion

Chapitre 9

Nouvelle approche pour l'interprétation de fonds documentaires

Ce dernier chapitre reprend les points essentiels de nos travaux, et présente quelques axes de recherche qui pourraient être explorés à présent.

9.1 Contribution à l'interprétation de fonds documentaires

9.1.1 Rappel des objectifs

Les fonds documentaires anciens et dégradés sont des données difficiles à traiter. Les systèmes existants se basent sur deux sources de connaissance pour interpréter les images : les connaissances extraites de ces dernières, et des connaissances à priori fournies par le concepteur. Toutefois, ces systèmes échouent à produire des résultats fiables face à l'incertitude, à la variabilité et au volume des fonds d'archives. De plus, leur conception peut devenir complexe dans ces circonstances.

Notre objectif a donc été de chercher comment permettre une diminution de l'effort manuel nécessaire à la correction des résultats produits par ces systèmes, sans augmenter la difficulté de leur conception.

Nous avons montré que de nouvelles sources de connaissance devaient être utilisées : le *contexte documentaire*, et les *opérateurs humains*, que l'on a proposé de regrouper sous le terme *« d'environnement »* lors de l'interprétation d'une page. Ces sources de connaissances doivent être utilisées *pendant les traitements*, pour permettre au système de prendre de meilleures décisions, et corriger au plus tôt ses erreurs.

Six contraintes fondamentales découlent de ces observations.

1. Un *mécanisme itératif* est nécessaire à la mise en place d'une interprétation *contextuelle* et *assistée*, afin de permettre un apport incrémental d'informations au système, et la réinterprétation des pages à mesure que de nouvelles connaissances sont disponibles. Ce mécanisme permet un échange à double sens entre une page et d'autres pages, mais aussi entre un module d'interprétation de page et un opérateur humain, à mesure que le système progresse dans l'interprétation.
2. Un *niveau global* doit permettre la gestion de la circulation d'informations entre les pages du fonds documentaire, et avec les opérateurs humains.

3. L'interaction entre les parties du systèmes doit être *asynchrone* pour éviter de faire attendre les opérateurs humains, ou bloquer les modules de traitement automatiques.

4. Au niveau de la page, l'information externe doit être *fusionnée* avec l'information extraite de l'image, afin de permettre la *remise en cause des résultats* produits précédemment, et la génération de nouveaux résultats plus fiables.

5. Afin de *minimiser la sollicitation des opérateurs humains*, il est nécessaire de permettre au système de *réclamer* l'information qui lui manque. Toutefois, dans la mesure où la mise en place de mécanismes de détection d'erreurs n'est pas toujours possible, il faut laisser aux opérateurs humains la possibilité de détecter *spontanément* les erreurs et les corriger.

6. La conception doit rester *simple et centralisée*. La description de la page ne doit pas être perturbée par des considérations temporelles ou des échanges d'information avec son environnement. Elle ne doit pas non plus être fragmentée entre plusieurs modules. La gestion de la circulation au sein du système doit donc être gérée automatiquement.

9.1.2 Points forts de nos travaux

Nous avons proposé une architecture et une méthode pour transformer un système existant et lui permettre de répondre aux besoins exprimés précédemment.

Le modèle de système que nous proposons permet au concepteur de s'appuyer, lors de la description du contenu de la page, sur des connaissances qui seront extraites globalement, sans avoir à savoir comment elles seront produites. Pour bénéficier de ces connaissances, il n'a qu'à exprimer :

- au niveau de la page : des propriétés à propos des contraintes de cohérence des contenus, et identifier les parties de la description impactées par les erreurs qu'il peut anticiper ;
- au niveau du fonds documentaire : un scénario décrivant la circulation de l'information entre les modules utilisés.

Au prix de cette perturbation minime de la description de page, notre mise en œuvre d'un mécanisme itératif global permet de gérer automatiquement :

- l'interruption de l'interprétation en cas d'erreur ;
- l'échange d'informations entre le module d'interprétation de page et son environnement ;
- la réintégration d'informations contextuelles au niveau de la page, leur validation, et la production de nouveaux résultats plus fiables.

Ces opérations, presque invisibles pour le concepteur, permettent une correction des erreurs, détectées automatiquement ou par un opérateur humain, au cours des différentes itérations du traitement global.

Notre approche permet donc, pour des scénarios bien écrits par le concepteur, une *diminution* de la quantité de travail manuel nécessaire pour atteindre un objectif de qualité donné, par rapport aux approches où les corrections sont effectuées après les traitements, et ce *sans augmenter* l'effort de conception nécessaire. La description de la page reste exempte de considérations temporelles, et unitaire. Grâce au niveau de décision matérialisé par le module de stratégie globale, il est également possible de gérer en un point unique la quantité de travail manuel qui sera nécessaire pour le traitement d'un fonds.

9.1.3 Intérêt pour la communauté scientifique

Pour la communauté scientifique du traitement de documents, nos propositions ont trois avantages majeurs.

Elles ont, tout d'abord, été *validées rigoureusement* à quatre niveaux :

Formel Nous avons montré au chapitre 6 la cohérence intrinsèque de nos travaux.

Pratique Nous avons montré au chapitre 7 la réalisation effective d'un système global et son application à des cas concrets.

Expérimental Nous avons montré au chapitre 8 la validation numérique de scénarios mise en œuvre à l'aide de nos outils sur des cas artificiels avec des données significatives : 1 637 vignettes de nombres, dans le cas du problème de sous-segmentation, et plus de 11 000 vignettes de patronymes, dans le cas du problème de transcription.

Productif Nous avons exploité les outils que nous avons réalisés pour construire un système capable de traiter les documents fournis par les Archives départementales des Yvelines, dans le cadre du partenariat que nous avons évoqué en introduction. Ceci représente, à l'heure où nous écrivons ces lignes, plus de 1 200 pages.

Nos propositions sont, par ailleurs, *faciles à transposer* dans un autre contexte. En effet, la formalisation de notre approche la rend indépendante d'un système existant, et nous avons montré qu'elle pouvait être appliquée pour étendre la méthode DMOS-P. Les mécanismes à la base de la mémoire visuelle sont mis en œuvre à l'aide de structures de données simples à reconstruire. L'extension du langage de description que nous avons proposés s'appuie quant à elle sur des mécanismes standards de continuation, qui sont disponibles dans la plupart des langages de programmation. La transformation d'un système existant en vue de le rentre interactif, et capable de s'intégrer dans une architecture globale itérative, peut donc être réalisée relativement simplement.

Finalement, notre architecture globale *facilite l'intégration de travaux* de diverses origines. Grâce à une communication asynchrone entre les modules du système, et au principe selon lequel le module d'interprétation de page ne fait pas d'hypothèse sur l'origine des données qu'il exploite, il est facile d'ajouter de nouveaux modules et de les composer pour réaliser des traitement globaux complexes. D'un point de vue plus pragmatique, cela permet d'utiliser simplement, au niveau du traitement de la page, des outils réalisés par plusieurs personnes.

9.2 Perspectives

9.2.1 Consolidation des propositions

Un premier prolongement possible de nos travaux consiste, assez naturellement, à consolider ces derniers.

9.2.1.1 Augmenter le niveau de remise en cause du système

Dans un système d'interprétation de fonds documentaires, il est souhaitable de pouvoir remettre en cause :

1. les résultats (intermédiaires ou finaux) liés à l'interprétation de la page : valeur d'un nombre, position d'une colonne, etc. ;

2. les paramètres numériques des modèles de pages : tailles de cases, hauteur moyenne de l'écriture, etc. ;
3. les lexiques et les contraintes linguistiques : dictionnaire de patronymes, liste des noms de communes, etc. ;
4. les modèles graphiques des contenus : modèles de caractères, modèles de symboles, etc. ;
5. les modèles structurels des pages : relations spatiales, grammaires bi-dimensionnelles, etc.

Nos propositions ont permis la mise en place du premier niveau de remise en cause, et certains de nos travaux sont encourageants pour la gestion du deuxième. Nous pensons en effet qu'il est envisageable d'exprimer des propriétés à propos des paramètres, dans la description de la page, de la même manière que pour les résultats intermédiaires. Nous pourrions par exemple, à l'aide d'une construction de la forme suivante, indiquer qu'une zone de positionnement est imprécise.

```
colonne_no_vente() ::=
   AT(IMPRECISE(PartieGauchePage)) && ...
```

Avec un tel opérateur, on pourrait collecter la position des éléments effectivement détectés, pour ensuite regrouper ces positions au niveau du fonds documentaire et déterminer des paramètres de positionnement plus précis.

Une expérience encourageante sur des registres matricules dégradés a été menée dans cette direction. Ces formulaires présentent une structure stable au sein d'un registre, mais certains éléments sont très dégradés. En démarrant avec une zone de recherche large, et des critères de validation strictes, il est possible d'extraire d'un registre quelques exemples de structures de formulaires bien reconnus. Grâce à un mécanisme permettant de préciser les paramètres de positionnement (un algorithme K-means par exemple), il est possible de réaliser un second traitement sur les pages en recherchant plus précisément les filets effacés, et en tolérant des données plus dégradées.

Dans la mesure où les mécanismes que nous avons proposés permettent de faciliter la production de quelques résultats fiables dans des contextes très difficiles, on peut espérer débloquer certains problèmes en s'appuyant sur des techniques d'apprentissages. Certains travaux dans cette direction ont d'ailleurs été démarrés dans notre équipe.

9.2.1.2 Cataloguer les circulations d'informations

La gestion de la remise en cause implique une bonne maîtrise de la circulation de l'information entre les modules, pour permettre la production de données capable d'influencer positivement et durablement le système.

Nous avons identifié certaines circulations typiques entre les modules pour les modèles de scénarios suivants :

- le regroupement d'éléments visuellement similaires ;
- la validation d'éléments peu fiables par un opérateur humain ;
- l'optimisation sur plusieurs pages d'une séquence d'éléments pour lesquels on dispose d'hypothèses de reconnaissance ;
- l'ajustement de la valeur d'un paramètre sur un ensemble de pages.

Une question intéressante est de savoir s'il est possible de définir un ensemble suffisant de circulations génériques, qui serait disponible au niveau du fonds documentaire, et de mettre en place des opérateurs de description, au niveau de la description de page, qui

permettent d'exploiter automatiquement une de ces circulations. Avec ce type d'approche, le concepteur n'aurait plus à spécifier de stratégie globale, et pourrait se concentrer sur l'expression de propriétés utiles à propos du contenu de la page.

9.2.2 Extension du champ d'application

Un second axe de travail possible est d'étendre le champ d'application de nos propositions. Deux directions sont alors envisageables.

9.2.2.1 Exploiter le système en consultation

Le modèle de système que nous avons proposé est destiné à l'étape d'extraction des connaissances, au cours du processus global de dématérialisation de fonds documentaires. En pratique, il est fréquent de recopier les données produites lors de la phase d'interprétation, une fois que celle-ci est terminée, vers un système dédié à la consultation des résultats.

L'utilisation de deux systèmes distincts est-elle nécessaire ? Ne peut-on pas, avec un même système, à la fois traiter les images et proposer d'utiliser les résultats alors produits ?

Ces question posent, implicitement, la question du rôle des utilisateurs finaux du système : sont-ils capables d'enrichir les données produites ? Serait-il possible de relancer certains traitements, après quelques temps, afin de faire progresser la connaissance globale du système à l'aide des informations saisies par les utilisateurs ?

Dans le mesure où le format de base de donnée que nous avons proposé semble adapté aux problèmes d'interprétation *et* d'indexation, il serait possible d'explorer de nouveaux usages à l'aide de nos outils.

9.2.2.2 Appliquer le mécanisme d'interprétation itératif à d'autres domaines

Pour terminer, on peut se poser la question du domaine d'application. Le mécanisme d'interprétation itératif que nous avons proposé permet la mise en place automatique d'un échange asynchrone d'informations entre un système de traitement et des opérateurs humains, grâce à l'expression de quelques propriétés à propos des données à interpréter.

Est-il possible de trouver, pour d'autres types de données structurées et en grand volume : un format pivot qui permet la fusion d'informations externes ; des propriétés utiles ; et des circulations d'informations productives ?

Un domaine avec lequel nous souhaiterions pouvoir confronter nos propositions est celui des pages Web. Variables et bruitées, elles peuvent pourtant être considérées par groupes cohérents : pages d'un même site, sites d'un même type de commerçant, etc. La modélisation d'un contexte, et la sollicitation maîtrisée d'opérateurs humains pourrait, peut-être, permettre l'extraction d'informations très structurées, à un coût maîtrisé, pour des volumes de données raisonnables.

Bibliographie

[1] S. Adam, M. Rigamonti, E. Clavier, E. Trupin, J.-M. Ogier, K. Tombre, and J. Gardes. Docmining : A document analysis system builder. In S. Marinai and A. Dengel, editors, *6th IAPR International Workshop on Document Analysis Systems - DAS 2004*, volume 3163 of *Lecture Notes in Computer Science*, pages 472–483, Florence, Italie, 2004. Springer Verlag.

[2] AFNOR. Description des manuscrits et fonds d'archives modernes et contemporains en bibliothèque : DeMArch, 2010. Dernière consultation le 3 août 2012, Accessible en ligne `http://www.bivi.fonctions-documentaires.afnor.org/livres-blancs/recommandation-de-description-des-manuscrits-et-fonds-d-archives`.

[3] A. Aho, R. Sethi, and J. Ullman. *Compilateurs : Principes, techniques et outils*. Dunod, 2000.

[4] A. Antonacopoulos and A. C. Downton. Special issue on the analysis of historical documents. *International Journal on Document Analysis and Recognition*, 9(2-4) :75–77, April 2007.

[5] H. Baird. Digital libraries and document image analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, volume 1, pages 2–14, Aug. 2003.

[6] H. S. Baird. Difficult and urgent open problems in document image analysis for libraries. In *DIAL '04 : Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, page 25, Washington, DC, USA, 2004. IEEE Computer Society.

[7] H. S. Baird, D. Lopresti, B. D. Davison, and W. M. Pottenger. Robust document image understanding technologies. In *HDP '04 : Proceedings of the 1st ACM workshop on Hardcopy document processing*, pages 9–14, New York, NY, USA, 2004. ACM.

[8] F. Bapst. *Reconnaissance de documents assistée : architecture logicielle et intégration de savoir-faire*. PhD thesis, Institut d'Informatique de l'Université de Fribourg, Suisse, 1998.

[9] F. Bapst, R. Brugger, and R. Ingold. Towards an interactive document structure recognition system. Internal working paper, 1995.

[10] F. Bapst, R. Brugger, A. Zramdini, and R. Ingold. L'intégration de données dans un système de reconnaissance de documents assistée. In *Actes du CNED'96*, 1996.

[11] F. Bapst, A. Zramdini, and R. Ingold. A scenario model advocating user-driven adaptive document recognition systems. *Proc. of ICDAR*, 0 :745, 1997.

[12] A. Belaïd, V. Poulain D'Andecy, H. Hamza, and Y. Belaïd. Administrative Document Analysis and Structure. In M. Biba and F. Xhafa, editors, *Learning Structure and Schemas from Documents*, volume 375 of *Studies in Computational Intelligence*, pages 51–72. Springer Verlag, Mar. 2011.

[13] C. Belleannée, P. Brisset, and O. Ridoux. A pragmatic reconstruction of lambdaprolog. *The Journal of Logic Programming*, 41(1) :67–102, 1999.

[14] T. M. Breuel. The ocropus open source ocr system. In *Proceedings IS&T/SPIE 20th Annual Symposium 2008*, 2008.

[15] J. Camillerapp. Utilisation des points d'intérêt pour rechercher des mots imprimés ou manuscrits dans des documents anciens. In *Conférence Internationale sur l'Ecrit et le Document (CIFED'12)*, pages 163–178, 2012.

[16] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, , and R. E. Gruber. Bigtable : A distributed storage system for structured data. In *OSDI'06 : Seventh Symposium on Operating System Design and Implementation*, 2006.

[17] J. Chazalon and B. Coüasnon. Using definite clause grammars to build a global system for analyzing collections of documents. In L. Likforman-Sulem and G. Agam, editors, *Document Recognition and Retrieval XVII*, volume 7534. SPIE, 2010.

[18] M. Cheriet, N. Kharma, C.-L. Liu, and C. Suen. *Character Recognition Systems : A Guide for Students and Practitioners*. John Wiley & Sons, 2007. 978-0-471-41570-1.

[19] E. Clavier, G. Masini, M. Delalandre, M. Rigamonti, K. Tombre, and J. Gardes. Docmining : A cooperative platform for heterogeneous document interpretation according to user-defined scenarios. In J. Lladós and Y.-B. Kwon, editors, *Graphics Recognition : Recent Advances and Perspectives - Selected papers from GREC'03 Fifth IAPR International Workshop on Graphics Recognition*, volume 3088 of *Lecture Notes in Computer Science*, pages 13–24. Springer Verlag, 2004.

[20] A. Cornuéjols, L. Miclet, and Y. Kodratoff. *Apprentissage artificiel - Concepts et algorithmes*. Algorithmes. Eyrolles, deuxième tirage 2003 edition, 2002. Langue : Français ISBN10 : 2-212-11020-0 ISBN13 : 978-2-212-11020-3 EAN13 : 9782212110203.

[21] G. Costagliola, A. De Lucia, S. Orefice, and G. Tortora. A parsing methodology for the implementation of visual systems. *Software Engineering, IEEE Transactions on*, 23(12) :777 –799, Dec. 1997.

[22] G. Costagliola, A. D. Lucia, S. Orefice, and G. Tortora. *Visual Language Theory*, chapter Positionnal Grammars : A Formalism for LR-like Parsing of Visual Languages, pages 171–191. Springer-Verlag, 1998.

[23] B. Coüasnon. *Segmentation et reconnaissance de documents guidées par la connaissance a priori : application aux partitions musicales*. Thèse de doctorat, Université de Rennes 1, France, 1996.

[24] B. Coüasnon. DMOS, a generic document recognition method : application to table structure analysis in a general and in a specific way. *International Journal on Document Analysis and Recognition*, 8(2–3) :111–122, June 2006.

[25] B. Coüasnon. *Fusion of Knowledge in Document Analysis*. PhD thesis, INSA de Rennes, 2012. Habilitation à Diriger des Recherches.

[26] B. Coüasnon, P. Brisset, and I. Stephan. Using logic programming languages for optical music recognition. In *In Proceedings of the Third International Conference on The Practical Application of Prolog*, pages 115–134, 1995.

[27] B. Coüasnon, J. Camillerapp, and I. Leplumey. Access by content to handwritten archive documents : generic document recognition method and platform for annotations. *International Journal on Document Analysis and Recognition*, 9(2–4) :223–242, April 2007.

[28] K. Coyle. Mass digitization of books. *The Journal of Academic Librarianship*, 32(6) :641 – 645, 2006.

[29] B. Coüasnon. Dmos : a generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 215 –220, 2001.

[30] B. Coüasnon. What can we learn from the processing of 165,000 forms from the 19th century ? In *Document Image Analysis for Libraries, 2006. DIAL '06. Second International Conference on*, 2006.

[31] J. Dean and S. Ghemawat. Mapreduce : Simplified data processing on large clusters. In *OSDI'04 : Sixth Symposium on Operating System Design and Implementation*, 2004.

[32] H. Déjean and J.-L. Meunier. Logical document conversion : combining functional and formal knowledge. In *DocEng '07 : Proceedings of the 2007 ACM symposium on Document engineering*, pages 135–143, New York, NY, USA, 2007. ACM.

[33] A. R. Dengel and B. Klein. smartfix : A requirements-driven system for document analysis and understanding. In *Document Analysis Systems V*, volume 2423/2002 of *Lecture Notes in Computer Science*, pages 77–88. Springer-Verlag, 2002.

[34] M. Droettboom, I. Fujinaga, K. MacMillan, G. S. Chouhury, T. DiLauro, M. Patton, and T. Anderson. Using the gamera framework for the recognition of cultural heritage materials. In *JCDL '02 : Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 11–17, New York, NY, USA, 2002. ACM.

[35] M. Droettboom, K. MacMillan, and I. Fujinaga. The gamera framework for building custom recognition systems. In *Symposium on Document Image Understanding Technologies*, pages 275–286, Marriott Hotel, Greenbelt, Maryland, April 2003.

[36] V. Eglin, F. Lebourgeois, S. Bres, H. Emptoz, Y. Leydier, I. Moalla, and F. Drira. Computer assistance for digital libraries : contributions to middle-ages and authors' manuscripts exploitation and enrichment. In *Document Image Analysis for Libraries, 2006. DIAL '06. Second International Conference on*, pages 16 pp. –280, april 2006.

[37] F. Esposito, D. Malerba, G. Semeraro, S. Ferilli, O. Altamura, T. Basile, M. Berardi, M. Ceci, and N. Di Mauro. Machine learning methods for automatically processing historical documents : from paper acquisition to xml transformation. In *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 328 – 335, 2004.

[38] C. Faure and N. Vincent. Document image analysis for active reading. In *SADPI '07 : Proceedings of the 2007 international workshop on Semantically aware document processing and indexing*, pages 7–14, New York, NY, USA, 2007. ACM.

[39] B. Forcher, S. Agne, A. Dengel, M. Gillmann, and T. Roth-Berghofer. Towards understandable explanations for document analysis systems. In *10th IAPR International Workshop on Document Analysis Systems*, 2012.

[40] R. Frost, R. Hafiz, and P. Callaghan. Parser combinators for ambiguous left-recursive grammars. In *Proceedings of the 10th International Symposium on Practical Aspects of Declarative Languages (PADL), ACM-SIGPLAN*, San Francisco, USA, January 2008.

[41] D. Q. Goldin. Persistent turing machines as a model of interactive computation. In *FoIKS '00 : Proceedings of the First International Symposium on Foundations of Information and Knowledge Systems*, pages 116–135, London, UK, 2000. Springer-Verlag.

[42] N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, and J.-C. Simon. A2ia check reader : A family of bank check recognition systems. *Document Analysis and Recognition, International Conference on*, 0 :523, 1999.

[43] L. Guichard, J. Chazalon, and B. Coüasnon. Exploiting Collection Level for Improving Assisted Handwritten Words Transcription of Historical Documents. In *International Conference on Document Analysis and Recognition (ICDAR'2011)*, pages 875–879, 2011.

[44] L. Guichard, A. Toselli, and B. Coüasnon. Un nouveau système indépendant de rejet multi-seuils pour la reconnaissance de mots manuscrits. In *Actes du 17ème Congrès Francophone de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA'10)*, 2010.

[45] L. Guichard, A. H. Toselli, and B. Coüasnon. Handwritten word verification by svm-based hypotheses re-scoring and multiple thresholds rejection. In *ICFHR*, pages 57–62, Kolkata, India, 2010.

[46] L. Guichard, A. H. Toselli, and B. Coüasnon. A novel verification system for handwritten words recognition. In *International Conference on Pattern Recognition (ICPR 2010)*, 2010.

[47] K. Hadjar, O. Hitz, L. Robadey, and R. Ingold. Configuration recognition model for complex reverse engineering methods : 2(crem). In *Document Analysis Systems*, pages 469–479, 2002.

[48] H. Hamza, Y. Belaïd, A. Belaïd, and B. Chaudhuri. An end-to-end administrative document analysis system. In *The Eighth IAPR International Workshop on Document Analysis Systems - DAS 2008*, pages 175–182, Nara Japon, 2008. IEEE.

[49] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *PAMI*, 31(1) :59–73, January 2009.

[50] J. Howard. Google begins to scale back its scanning of books from university libraries. *The Chronicle of Higher Education*, March, 9 2012. Available online at `http://chronicle.com/article/Google-Begins-to-Scale-Back/131109/`.

[51] S. L. Huitouze, P. Louvet, and O. Ridoux. Les grammaires logiques et λProlog. In *Journées Francophones sur la Programmation en Logique*, pages 93–108, Nîmes, France, 1993. Teknea.

[52] B. Klein, S. Agne, and A. D. Bagdanov. Understanding document analysis and understanding (through modeling). In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 2, page 1218, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[53] B. Klein, A. Dengel, and A. Fordan. smartfix : An adaptive system for document analysis and understanding. In A. Dengel, M. Junker, and A. Weisbecker, editors, *Reading and Learning*, volume 2956/2004 of *Lecture Notes in Computer Science*, pages 166–186. Springer Berlin / Heidelberg, 2004.

[54] V. Kluzner, A. Tzadok, Y. Shimony, E. Walach, and A. Antonacopoulos. Word-based adaptive ocr for historical books. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 501 –505, july 2009.

[55] D. E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2) :127–145, 1968.

[56] B. Lamiroy and D. Lopresti. The non-geek's guide to the dae platform. In *10th IAPR International Workshop on Document Analysis Systems*, 2012.

[57] B. Lazzerini, F. Marcelloni, and L. M. Reyneri. Beatrix : A self-learning system for off-line recognition of handwritten texts. *Pattern Recognition Letters*, 18(6) :583 – 594, 1997.

[58] F. LeBourgeois, S. Souafi-Bensafi, J. D. J, M. Parizeau, M. Coté, and H. Emptoz. Using statistical models in document images understanding. In *DLIA'01*, 2001.

[59] A. Lemaitre, J. Camillerapp, and B. Couasnon. Contribution of multiresolution description for archive document structure recognition. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 247 –251, sept. 2007.

[60] A. Lemaitre, J. Camillerapp, and B. Coüasnon. Multi-script baseline detection using perceptive vision. In *14th Biennial Conference of the International Graphonomics Society (IGS 2009)*, 2009.

[61] A. Lemaitre-Legargeant. *Introduction de la vision perceptive pour la reconnaissance de la structure de documents*. PhD thesis, Institut National des Sciences Appliquées de Rennes (INSA), France, 2008.

[62] M. Lemaître, E. Grosicki, E. Geoffrois, and F. Prêteux. Layout analysis of handwritten letters based on textural and spatial information and a 2d markovian approach. In *Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2008.

[63] C. C. Lin, Y. Niwa, and S. Narita. Logical structure analysis of book document images using contents information. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 2, pages 1048–1054, Aug 1997.

[64] J. Lladós, D. Karatzas, J. Mas, and G. Sánchez. A generic architecture for the conversion of document collections into semantically annotated digital archives. *Journal of Universal Computer Science*, 14(18) :2912–2935, oct 2008.

[65] J. Lladós, E. Valveny, G. Sánchez, and E. Martí. *Symbol Recognition : Current Advances and Perspective*, volume 2390/2002 of *Graphics Recognition Algorithms and Applications, Lecture Notes in Computer Science*, pages 104–128. Springer Berlin / Heidelberg, 2002.

[66] G. Lorette. Handwriting recognition or reading ? what is the situation at the dawn of the 3rd millenium ? *International Journal on Document Analysis and Recognition*, 2(1) :2–12, July 1999.

[67] H. Ma and D. S. Doermann. Bootstrapping structured page segmentation. In *Proc. SPIE 5010*, volume 179 of *Document Recognition and Retrieval X*, 2003.

[68] S. Macé and E. Anquetil. Eager interpretation of on-line hand-drawn structured documents : The dali methodology. *PR*, 2008.

[69] J. V. Mahoney and M. P. J. Fromherz. Three main concerns in sketch recognition and an approach to addressing them. In *AAAI Spring Symposium on Sketch Understanding*. Palo Alto Research Center, Xerox, 2002.

[70] D. Malerba, F. Esposito, and O. Altamura. Learning rules for layout analysis correction. In *DLIA 2001 Advance Program*, 2001.

[71] D. Malerba, F. Esposito, O. Altamura, M. Ceci, and M. Berardi. Correcting the document layout : A machine learning approach. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 1*, ICDAR '03, pages 97–, Washington, DC, USA, 2003. IEEE Computer Society.

[72] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms : a literature survey. In *Document Recognition and Retrieval X, (Proceedings of SPIE/IST)*, volume 5010, Santa Clara, California, January 2003.

[73] A. O. Maroneze, B. Coüasnon, and A. Lemaitre. Introduction of statistical information in a syntactic analyser for document image recognition. In *Document Recognition and Retrieval XVII*, 2011.

[74] K. Marriott and B. Meyer, editors. *Visual Language Theory*. Springer-Verlag, 1998.

[75] K. Marriott and B. Meyer. *Visual Language Theory*, chapter The CCMG Visual Language Hierarchy, pages 129–169. Springer-Verlag, 1998.

[76] K. Marriott, B. Meyer, and K. B. Wittenburg. *Visual Language Theory*, chapter A Survey of Visual Language Specification and Recognition, pages 5–85. Springer-Verlag, 1998.

[77] I. Martinat and B. Coüasnon. A minimal and sufficient way of introducing external knowledge for table recognition in archival documents. In W. Liu and J. Lladós,

editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 206–217. Springer-Verlag, 2006.

[78] D. Michie. Memo functions and machine learning. *Nature*, 218(5136) :19–22, Apr. 1968.

[79] F. Montreuil, E. Grosicki, L. Heutte, and S. Nicolas. Unconstrained handwritten document layout extraction using 2d conditional random fields. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 853 –857, july 2009.

[80] G. Nagy. Teaching a computer to read. In *Pattern Recognition, 1992. Vol.II. Conference B : Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 225–229, Sept. 1992.

[81] G. Nagy. Twenty years of document image analysis in pami. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :38–62, 2000.

[82] G. Nagy and S. Veeramachaneni. Adaptive and interactive approaches to document analysis. In S. Marinai and H. Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 221–257. Springer, 2008.

[83] A. M. Namboodiri and A. Jain. *Digital Document Processing : Major Directions and Recent Advances*, chapter Document Structure and Layout Analysis, pages 29–48. Springer-Verlag, London, Jan. 2007. (ISBN :978-1-84628-501-1).

[84] H. P. Nii. Blackboard systems. Technical Report STAN-CS-86-1123 (also numbered KSL 86-18), Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford University, Stanford, CA 94305, June 1986.

[85] J. Ogier, R. Mullot, J. Labiche, and Y. Lecourtier. Semantic coherency : the basis of an image interpretation device-application to the cadastral map interpretation. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 30(2) :322 –338, apr 2000.

[86] J.-M. Ogier. Ancient document analysis : A set of new research problems. In *Actes du dixième Colloque International Francophone sur l'Écrit et le Document*, pages 73–78, 2007.

[87] J. M. Ogier, R. Mullot, J. Labiche, and Y. Lecourtier. Multilevel approach and distributed consistency for technical map interpretation : Application to cadastral maps. *Computer Vision and Image Understanding*, 70(3) :438 – 451, 1998.

[88] S. Orefice, G. Polese, M. Tucci, G. Tortora, G. Costagliola, and S. Chang. A 2d interactive parser of iconic languages. In *Proc. IEEE Workshop on Visual Languages*, pages 207–213, Sep 1992.

[89] L. O'Gorman and R. Kasturi. *Document image analysis*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[90] J. Paakki. Attribute grammar paradigms—a high-level methodology in language implementation. *ACM Comput. Surv.*, 27(2) :196–255, 1995.

[91] B. Pasternak. Processing imprecise and structural distorted line drawings by an adaptable drawing interpretation kernel. In *Proceedings of IAPR Workshop on Document Analysis Systems (DAS)*, pages 349–365, 1994.

[92] B. Pasternak. The role of taxonomy in drawing interpretation. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 799 –802 vol.2, aug 1995.

[93] B. Pasternak, G. Gabrielides, and R. Sprengel. Wiz — a prototype for knowledge-based drawing interpretation. In F. Belli and F. Radermacher, editors, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 604 of *Lecture Notes in Computer Science*, pages 164–173. Springer Berlin / Heidelberg, 1992. 10.1007/BFb0024968.

[94] B. Pasternak and B. Neumann. Adaptable drawing interpretation using object-oriented and constraint-based graphic specification. In *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 359 –364, oct 1993.

[95] F. C. N. Pereira and D. H. D. Warren. Definite clause grammars for language analysis–a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3) :231 – 278, 1980.

[96] R. Plamondon and S. Srihari. Online and off-line handwriting recognition : a comprehensive survey. *PAMI*, 22(1) :63 – 84, January 2000.

[97] J. Ramel, S. Busson, and M. Demonet. Agora : the interactive document image analysis tool of the bvh project. *Document Image Analysis for Libraries, International Workshop on*, 0 :145–155, 2006.

[98] J. Ramel and S. Leriche. Segmentation et analyse interactives de documents anciens imprimés. In *CIFED Colloque International Francophone sur l'Ecrit et le Document No7*, 2004.

[99] J.-Y. Ramel. *Propositions pour la représentation et l'analyse de documents numériques*. Habilitation à diriger des recherches, Université François Rabelais de Tours, November 2006.

[100] J. Y. Ramel, S. Leriche, M. L. Demonet, and S. Busson. User-driven page layout analysis of historical printed books. *International Journal of Document Analysis and Recognition*, 9(2-3) :243–261, April 2007.

[101] T. M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *International Conference on Document Analysis and Recognition*, volume 1, page 218, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[102] O. Ridoux. *λProlog de A a Z . . . ou presque*. Habilitation à diriger des recherches, Université de Rennes 1, 1998.

[103] L. Robadey. *2(CREM) : Une méthode de reconnaissance structurelle de documents complexes basée sur des patterns bidimensionnels*. PhD thesis, Faculté des Sciences de l'Université de Fribourg (Suisse), 2001.

[104] L. Robadey, O. Hitz, and R. Ingold. A pattern-based method for document structure recognition. In *DLIA'01*, 2001.

[105] Y. Saidali, S. Adam, J. M. Ogier, E. Trupin, and J. Labiche. Knowledge representation and acquisition for engineering document analysis. In J. Lladós and Y.-B. Kwon, editors, *Graphics Recognition*, volume 3088 of *Lecture Notes in Computer Science*, pages 25–37. Springer Berlin / Heidelberg, 2004.

[106] E. Saund. Scientific challenges underlying production document processing. In *Document Recognition and Retrieval XVIII*, volume 7874-1 of *Proceedings of SPIE*, 2011.

[107] K. M. Sayre. Machine recognition of handwritten words : A project report. *Pattern Recognition*, 5(3) :213 – 228, 1973.

[108] L. Schomaker. Reading systems : An introduction to digital document processing. In B. B. Chaudhuri, editor, *Digital Document Processing*, Advances in Pattern Recognition, pages 1–28. Springer London, 2007.

[109] D. Scott and C. Strachey. Toward A Mathematical Semantics for Computer Languages. In J. Fox, editor, *Proceedings of the Symposium on Computers and Automata*, volume XXI, pages 19–46, Brooklyn, N.Y., Apr. 1971. Polytechnic Press.

[110] B. Seidler, M. Ebbecke, and M. Gillmann. smartfix statistics : towards systematic document analysis performance evaluation and optimization. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, pages 333–340, New York, NY, USA, 2010. ACM.

[111] F. Shafait, D. Keysers, and T. Breuel. Performance evaluation and benchmarking of six-page segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6) :941 –954, june 2008.

[112] F. Shafait, D. Keysers, and T. M. Breuel. Performance comparison of six algorithms for page segmentation. In *in 7th IAPR Workshop on Document Analysis Systems*, pages 368–379. Springer, 2006.

[113] Y. Tang, C. Suen, C. D. Yan, and M. Cheriet. Financial document processing based on staff line and description language. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(5) :738–754, May 1995.

[114] Y. Y. Tang, S.-W. Lee, and C. Y. Suen. Automatic document processing : A survey. *Pattern Recognition*, 29(12) :1931 – 1952, 1996.

[115] Y. Y. Tang, C. D. Yan, and C. Suen. Document processing for automatic knowledge acquisition. *Knowledge and Data Engineering, IEEE Transactions on*, 6(1) :3–21, Feb 1994.

[116] A. Toselli, V. Romero, E. Vidal, and L. Rodriguez. Computer assisted transcription of handwritten text images. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 944 –948, sept. 2007.

[117] A. H. Toselli, V. Romero, and E. Vidal. Computer assisted transcription of text images and multimodal interaction. In *Machine Learning for Multimodal Interaction*, volume 5237/2008 of *Lecture Notes in Computer Science*, pages 296–308. Springer Berlin / Heidelberg, 2008.

[118] G. T. Toussaint. The use of context in pattern recognition. *Pattern Recognition*, 10(3) :189 – 204, 1978.

[119] P. Vaxivière and K. Tombre. Celesstin : Cad conversion of mechanical drawings. *Computer*, 25 :46–54, 1992.

[120] E. Vidal, L. Rodríguez, F. Casacuberta, and I. García-Varea. Interactive pattern recognition. In A. Popescu-Belis, S. Renals, and H. Bourlard, editors, *Machine Learning for Multimodal Interaction*, volume 4892 of *Lecture Notes in Computer Science*, pages 60–71. Springer Berlin / Heidelberg, 2008.

[121] A. Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35(7) :1433 – 1446, 2002.

[122] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6) :709–720, 2004.

[123] T. Watanabe, Q. Luo, and N. Sugie. Structure recognition methods for various types of documents. *Machine Vision and Applications*, 6 :163–176, 1993. 10.1007/BF01211939.

[124] P. Wegner. Interactive foundations of computing. *Theoretical Computer Science*, 192(2) :315 – 351, 1998.

[125] P. Wegner and D. Goldin. Computation beyond turing machines. *Commun. ACM*, 46(4) :100–102, 2003.

[126] P. Xiu and H. Baird. Whole-book recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99) :1, 2012.

[127] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3) :44–49, Sept. 2005.

Résumé

Les fonds d'archives forment de grandes quantités de documents difficiles à interpréter automatiquement : les approches classiques imposent un lourd effort de conception, sans parvenir à empêcher la production d'erreurs qu'il faut corriger après les traitements.

Face à ces limites, notre travail vise à améliorer la processus d'interprétation, en conservant un fonctionnement page par page, et en lui apportant des informations contextuelles extraites du fonds documentaire ou fournies par des opérateurs humains.

Nous proposons une extension ciblée de la description d'une page qui permet la mise en place systématique d'échanges entre le processus d'interprétation et son environnement. Un mécanisme global itératif gère l'apport progressif d'informations contextuelles à ce processus, ce qui améliore l'interprétation.

L'utilisation de ces nouveaux outils pour le traitement de documents du XVIII^{e} siècle a montré qu'il était facile d'intégrer nos propositions à un système existant, que sa conception restait simple, et que l'effort de correction pouvait être diminué.

Abstract

Fonds, also called historical document collections, are important amounts of digitized documents which are difficult to interpret automatically: usual approaches require a lot of work during design, but do not manage to avoid producing many errors which have to be corrected after processing.

To cope with those limitations, our work aimed at improving the interpretation process by making use of information extracted from the fonds, or provided by human operators, while keeping a page by page processing.

We proposed a simple extension of page description language which permits to automatically generate information exchange between the interpretation process and its environment. A global iterative mechanism progressively brings contextual information to the later process, and improves interpretation.

Experiments and application of those new tools for the processing of documents from the 18th century showed that our propositions were easy to integrate in an existing system, that its design is still simple, and that required manual corrections were reduced.

N° d'ordre : **D13-01**

Institut National des Sciences Appliquées de Rennes

20, Avenue des Buttes de Coësmes . CS 14315 . F-35043 Rennes Cedex

Tel : 02 23 23 82 00 – Fax : 02 23 23 83 96